**ONLINE FOOD DELIVERY**

Abstract:

Online food delivery services have rapidly transformed the culinary and dining landscape over the past decade. This service model, which allows consumers to order food from a variety of local restaurants through a digital platform, has experienced significant growth due to advancements in technology, changing consumer preferences, and the global shift towards convenience-oriented lifestyles. Platforms such as Uber Eats, DoorDash, and Grubhub have become household names, providing a seamless and efficient way for customers to enjoy their favorite meals without leaving their homes. This introduction explores the evolution, mechanics, and impact of online food delivery services, highlighting their benefits, challenges, and future prospects.

Introduction:

The emergence of online food delivery services represents a significant shift in the food and beverage industry. Initially rooted in traditional takeaway models, these services have evolved to offer a comprehensive and sophisticated approach to dining, leveraging the power of digital technology. The basic premise involves a customer placing an order through a website or mobile app, which is then prepared by a partnered restaurant and delivered by a network of couriers.

The rapid proliferation of smartphones and increased internet penetration have been pivotal in the rise of online food delivery platforms. These services cater to the growing demand for convenience, enabling consumers to access a wide variety of cuisines from the comfort of their homes. Furthermore, the COVID-19 pandemic has accelerated this trend, as lockdowns and social distancing measures led to a surge in demand for home-delivered meals.

Function Requirements:

Functional requirements in UML (Unified Modeling Language) diagrams are depicted through various types of diagrams that each serve different purposes in modeling the behavior and structure of a system.

**Use Case Diagram**:

* **Purpose**: To capture the functional requirements of a system from an end-user perspective.
* **Elements**:
  + **Actors**: Represent users or other systems that interact with the system.
  + **Use Cases**: Represent the functional requirements (i.e., what the system should do).
  + **Relationships**: Include associations, extensions, and inclusions between use cases and actors.

**Activity Diagram**:

* **Purpose**: To model the workflow and the sequence of activities in a system.
* **Elements**:
  + **Activities**: Represent individual tasks or functions.
  + **Transitions**: Show the flow from one activity to another.
  + **Decision Points**: Represent branching based on conditions.
  + **Start and End Nodes**: Indicate the beginning and end of the process.

**Sequence Diagram**:

* **Purpose**: To illustrate how objects interact in a particular sequence to achieve a function.
* **Elements**:
  + **Objects/Participants**: Represent the different entities involved.
  + **Messages**: Show the communication between objects.
  + **Lifelines**: Represent the lifespan of an object during the interaction.
  + **Activation Bars**: Indicate when an object is active or processing.

**State Machine Diagram**:

* **Purpose**: To model the states an object goes through in response to events.
* **Elements**:
  + **States**: Represent different states of an object.
  + **Transitions**: Show the movement from one state to another.
  + **Events**: Trigger transitions between states.

**Class Diagram**:

* **Purpose**: To model the static structure of the system by showing its classes, attributes, operations, and relationships.
* **Elements**:
  + **Classes**: Represent the objects or concepts.
  + **Attributes**: Define the properties of a class.
  + **Operations**: Define the behaviors or functions of a class.
  + **Relationships**: Include associations, generalizations, and dependencies between classes.

Exception & Utility Class:

**1)Customer class**

1. public int customerID(PK)

2. public string Name

3. public String address

4. public string phone\_number

Define a parameterized constructor in the Customer to initialize values to the above fields.

**Constructor:** Customer(int customerId,String name,String address,String phone\_number)

**Exception class:**

CustomerNotFoundException()

InvalidCustomerInputException()

**CustomerUtility class** with methods:

**public void createCustomer(Customer customer) throws InvalidCustomerInputExceptionpublic void AddCustomer(Customer obj)**

**public** Customer readCustomer(**int** customerId) **throws** CustomerNotFoundException

**public** **void** updateCustomer(Customer customer) **throws** CustomerNotFoundException, InvalidCustomerInputException

**public** **void** deleteCustomer(**int** customerId) **throws** CustomerNotFoundException

**2)MenuClass:**

Long restaurantId;(FK)

String name;

String description;

Double price;

**Parametrized constructor** : Menu(String name,String description,Double price)

**Exception Class:**

public class MenuItemNotFoundException extends OnlineFoodDeliveryException

public MenuItemUnavailableException(String itemName)

**Menu Utility class:**

We use hashmap to store store id as key along with menu object.

public Menu addMenuItem(Menu menu)

public Menu getMenuItem(Long restaurantId)

public Menu updateMenuItem(Long restaurantId, Menu updatedMenu)

public void deleteMenuItem(Long restaurantId)

public void checkMenuItemAvailability(Long restaurantId)

public Map<Long, Menu> getAllMenuItems()

**3) Restaurant class:**

private int restaturantId;

            private String name;

            private String location;

            private String cusine;

            private String contact;

            private float rating;

**Constructor :** Restaurant(int id,String name,String location,String cusine,String contact,float rating)

**Exception Class**

public class RestaurantClosedException extends OnlineFoodDeliveryException

public class RestaurantNotFoundException extends OnlineFoodDeliveryException

**RestaurantUtility Class**

The RestaurantUtility class will provide helper methods for managing the list of restaurants.

public static void addRestaurant(Restaurant restaurant)

public static void displayAllRestaurants()

public static Restaurant findRestaurantById(int id) throws RestaurantNotFoundException

public static void displayRestaurantDetails(Restaurant restaurant)

**4)Order class:**

Private int orderId;

Private int customerId;(FK)

Private int restaurantId;(FK)

Private String deliveryAddress:

Private String double totalPrice;

Private String String status;

**Constructor:** order(int orderId,String deliveryAddress,double totalPrice,String status)

* Both customerId,restaurantId are fields are inherited from customer,restaurant classes

**Exception class:**

This exception used to invalid order data.

public class OrderNotFoundException extends OnlineFoodDeliveryException

public class InvalidOrderStateException extends OnlineFoodDeliveryException

**OrderUtility class:**

public static Order findOrderById(int orderId, List<Order> orders) throws OrderNotFoundException

public static void validateOrder(Order order) throws InvalidOrderStateException

**5)Payment class:**

Private int paymentId;

Private int orderId;(FK)

Private double amount;

Private String payment\_method;

**Constructor:**Payment(int paymentId,double amount,String payment\_method)

* orderId are inherited from order class.

**Exception class:**

This throws exception when invalid payment will happen.

public class PaymentFailedException extends OnlineFoodDeliveryException

public class InsufficientFundsException extends OnlineFoodDeliveryException

**PaymentUtility class:**

public static void processPayment(Payment payment, double availableFunds) throws PaymentFailedException, InsufficientFundsException

**6)Rating class:**

Private int ratingId;

Private int orderId;(FK)

Private int stars;

Private String comment;

**Constructor:**Rating(int ratingId,int stars,String comment)

* orderId are inherited from order class.

**Exception Class**

custom exception class for handling issues related to the rating, such as invalid stars or missing comments.

public class InvalidRatingException extends OnlineFoodDeliveryException

**RatingUtility Class**

**public static void validateComment(String comment) throws RatingException**

Method to validate that the comment is not empty

**public static void processRating(Rating rating)**

Method to simulate processing the rating

**public static void validateStars(int stars) throws RatingException**

Method to validate the number of stars (it must be between 1 and 5)

**7)Driver class:**

Private int driverId;

Private String name;

Private String phoneNumber;

Private String vehicleType;

**Constructor:**Driver(int driverId,String name,String phoneNumber,String vehicleType)

* restaurantId are inherited from restaurant class;

**Exception Class:**

public class DriverNotFoundException extends OnlineFoodDeliveryException

public class DriverUnavailableException extends OnlineFoodDeliveryException

**DriverUtility Class:**

public void createDriver(Driver driver) throws DriverUnavailableException

public Driver readDriver(int driverId) throws DriverNotFoundException

public void updateDriver(Driver driver) throws DriverNotFoundException, DriverUnavailableException

public void deleteDriver(int driverId) throws DriverNotFoundException

**8)Promotion class:**

Private int promotionId;

Private int restaurantId;(FK)

Private String promotionDetails;

Private String startDate;

Private String endDate;

**Constructor:** public Promotion(int restaurantId, String restaurantName, String restaurantAddress, int promotionId, String promotionDetails, String startDate, String endDate)

**PromotionException Class:**

A custom exception class for handling issues related to the promotion, such as invalid dates or missing promotion details.

public class PromotionExpiredException extends OnlineFoodDeliveryException

public class PromotionNotFoundException extends OnlineFoodDeliveryException

**PromotionUtility Class:**

public static void validateDates(String startDate, String endDate) throws PromotionExpiredException

//Method to validate that the start date is before the end date

public static void validatePromotionDetails(String promotionDetails) throws PromotionNotFoundException

//Method to validate that promotion details are not empty

public static void applyPromotion(Promotion promotion)

//Method to simulate applying a promotion to a restaurant

**9)Delivery class:**

Private int deliveryId;

Private int orderId;(FK)

Private int driverId;

Private String deliveryStatus;

**Constructor:**Delivery(deliveryId,driverId,deliveryStatus)

**Exception Class:**

A custom exception class to handle errors related to deliveries, such as invalid status or missing driver details.

public class DeliveryDelayedException extends OnlineFoodDeliveryException

public class DeliveryUnavailableException extends OnlineFoodDeliveryException

**DeliveryUtility Class:**

**public static void validateDeliveryStatus(String deliveryStatus) throws DeliveryException**

Method to validate delivery status (supports "Pending", "In Transit", "Delivered")

**public static void assignDriverToDelivery(Delivery delivery) throws DeliveryException**

Method to assign a driver to a delivery

**public static void trackDelivery(Delivery delivery)**

Method to simulate delivery tracking.

**Inheritance:**

**User Class (Base Class)**

* Shared by Customer and Driver.
* Contains general attributes like userId, name, and contactInfo.

**Customer Class (Inherits from User)**

* Contains additional attributes like address.

**Driver Class (Inherits from User)**

* Contains additional attributes like vehicle.

**BusinessEntity Class (For entities that represent businesses like Restaurant and Promotion)**

* General attributes like entityId, name, and contactInfo.

**Restaurant Class (Inherits from BusinessEntity)**

* Specific to restaurants, containing attributes like location and menu.
* Restaurant has a relationship with Menu.

**Menu Class (Belongs to a Restaurant)**

* Contains menuItems and is associated with a Restaurant.

**Order Class (Manages order information)**

* Central entity connecting Customer, Restaurant, and Driver.
* Has relationships with Delivery, Payment, and Rating.

**Delivery Class (Handles delivery information)**

* Associated with Order and Driver.

**Payment Class (Handles payment information)**

* Associated with Order.

**Rating Class (Handles ratings for orders, restaurants, or drivers)**

* Associated with Order, Restaurant, or Driver.

**Promotion Class (Belongs to BusinessEntity)**

* Can apply to either Customer or Restaurant

//Base User class

class User {

protected int userId;

protected String name;

protected String phoneNumber;

public User(int userId, String name, String phoneNumber) {

this.userId = userId;

this.name = name;

this.phoneNumber= phoneNumber;

}

}

// Customer class inheriting from User

class Customer extends User {

private String address;

public Customer(int userId, String name, String phoneNumber, String address) {

super(userId, name, phoneNumber);

this.address = address;

}

}

// Driver class inheriting from User

class Driver extends User {

private String vehicleType;

public Driver(int userId, String name, String phoneNumber, String vehicleType) {

super(userId, name, contactInfo);

this.vehicleType = vehicleType;

}

}

// Base BusinessEntity class

class BusinessEntity {

protected int entityId;

protected String name;

protected String contactInfo;

public BusinessEntity(int entityId, String name, String contactInfo) {

this.entityId = entityId;

this.name = name;

this.contactInfo = contactInfo;

}

}

// Restaurant class inheriting from BusinessEntity

class Restaurant extends BusinessEntity {

private String location;

private Menu menu;

public Restaurant(int entityId, String name, String contactInfo, String location) {

super(entityId, name, contactInfo);

this.location = location;

this.menu = new Menu();

}

public Menu getMenu() {

return menu;

}

}

// Menu class associated with a Restaurant

class Menu {

private List<String> menuItems;

public Menu() {

this.menuItems = new ArrayList<>();

}

public void addMenuItem(String item) {

menuItems.add(item);

}

public List<String> getMenuItems() {

return menuItems;

}

}

// Order class that links Customer, Restaurant, and Driver

class Order {

private int orderId;

private Customer customer;

private Restaurant restaurant;

private Driver driver;

public Order(int orderId, Customer customer, Restaurant restaurant, Driver driver) {

this.orderId = orderId;

this.customer = customer;

this.restaurant = restaurant;

this.driver = driver;

}

}

// Delivery class

class Delivery {

private int deliveryId;

private Driver driver;

private String deliveryTime;

public Delivery(int deliveryId, Driver driver, String deliveryTime) {

this.deliveryId = deliveryId;

this.driver = driver;

this.deliveryTime = deliveryTime;

}

}

// Payment class

class Payment {

private int paymentId;

private Order order;

private String paymentType;

public Payment(int paymentId, Order order, String paymentType) {

this.paymentId = paymentId;

this.order = order;

this.paymentType = paymentType;

}

}

// Rating class

class Rating {

private int ratingId;

private int ratingValue;

private String comments;

public Rating(int ratingId, int ratingValue, String comments) {

this.ratingId = ratingId;

this.ratingValue = ratingValue;

this.comments = comments;

}

}

// Promotion class inheriting from BusinessEntity

class Promotion extends BusinessEntity {

private double discount;

public Promotion(int entityId, String name, String contactInfo, double discount) {

super(entityId, name, contactInfo);

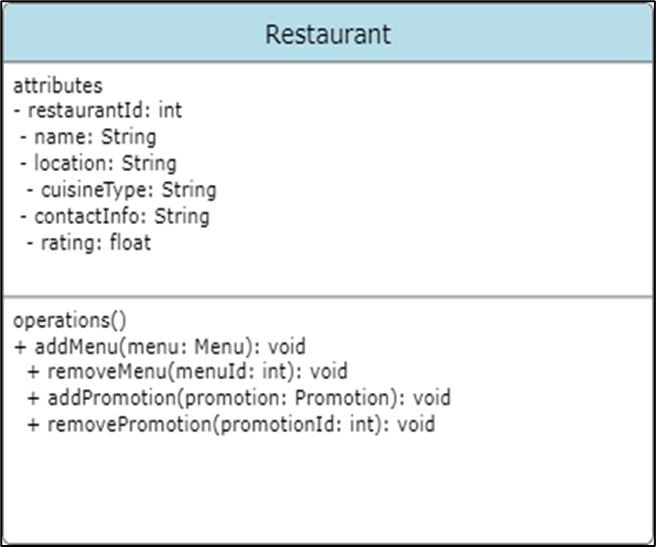
this.discount = discount;

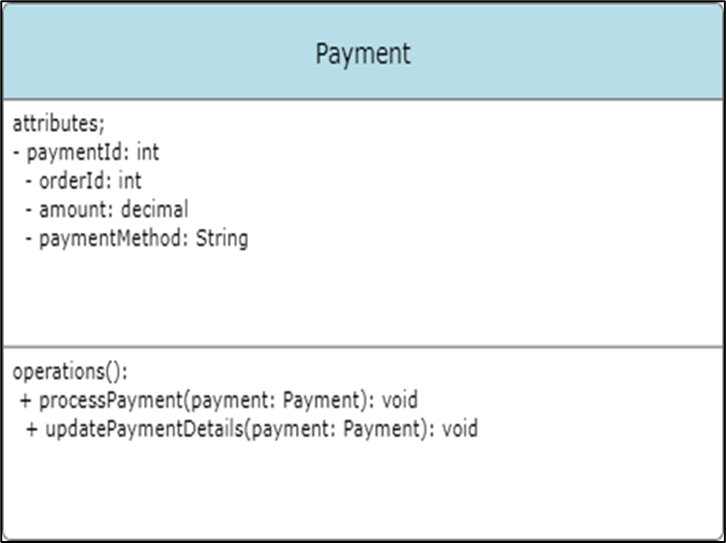
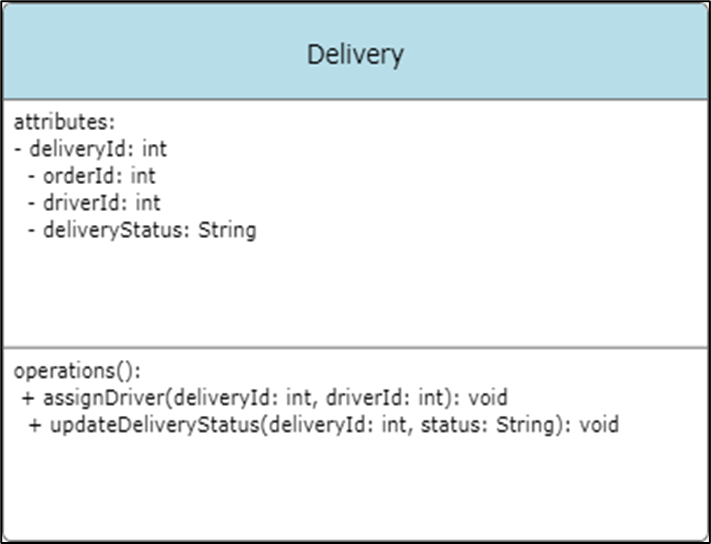
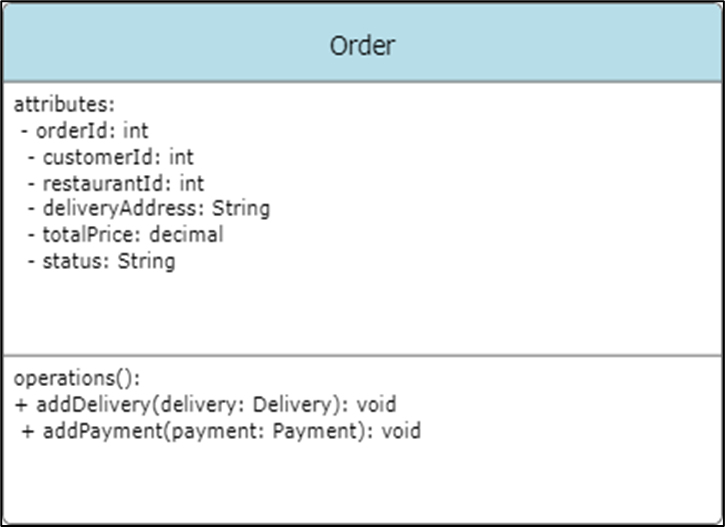
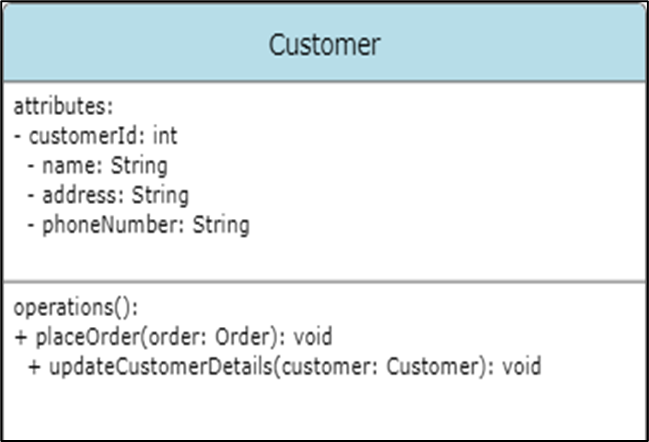
}

}

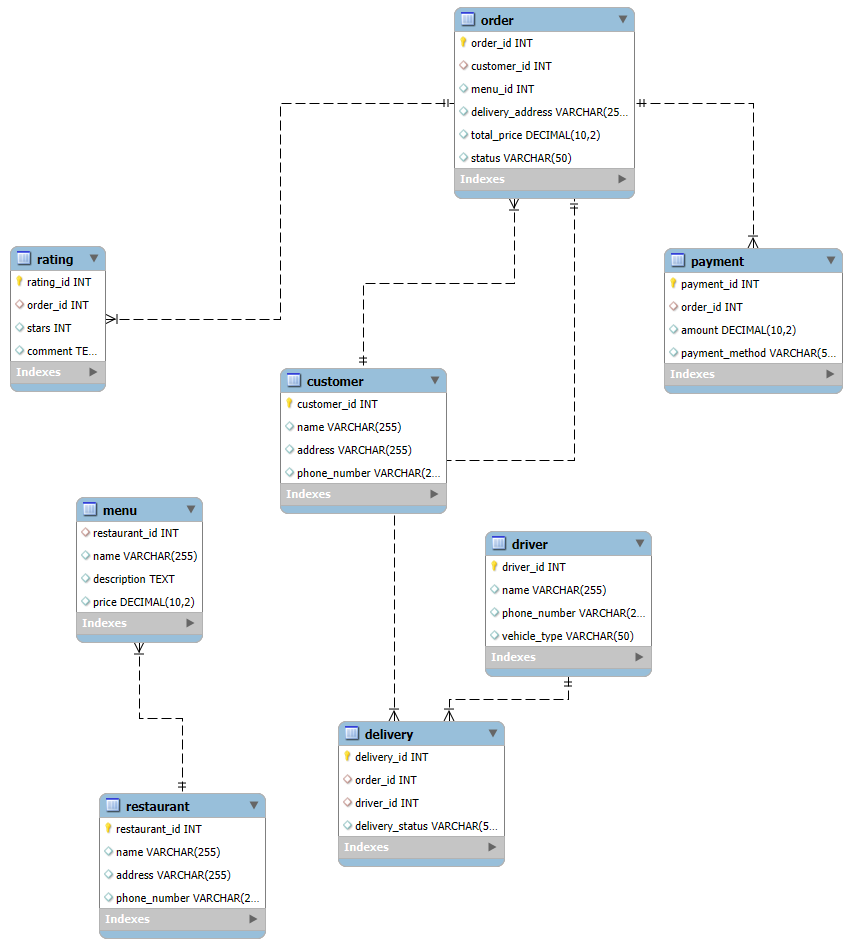
Class Diagram:

UML/CLASS DIAGRAM





ER DIAGRAMS:



DB Queries:

CREATE DATABASE foodweb;

USE foodweb;.

1.CREATE TABLE `customer` (

  `customer\_id` int NOT NULL,

  `customer\_name` varchar(255) DEFAULT NULL,

  `address` varchar(255) DEFAULT NULL,

  `phone\_number` varchar(20) DEFAULT NULL,

  PRIMARY KEY (`customer\_id`)

);

INSERT INTO `customer` VALUES

(1,'sunny','Mothkur','123-456-7890'),

(2,'saiganesh','saroornagar','234-567-8901'),

(3,'ganesh','jagitial','345-678-9012'),

(4,'ranjith','nakrekal','456-789-0123'),

(5,'sameer','uppal','567-890-1234');

Select \* from customer;

use foodweb;

2.CREATE TABLE foodweb.Delivery (

    delivery\_id INT PRIMARY KEY,

    order\_id INT,

    driver\_id INT,

    delivery\_status VARCHAR(50),

    FOREIGN KEY (order\_id) REFERENCES foodweb.Order(order\_id),

    FOREIGN KEY (driver\_id) REFERENCES foodweb.Driver(driver\_id)

);

INSERT INTO foodweb.Delivery (delivery\_id, order\_id, driver\_id, delivery\_status)

VALUES

    (41, 1, 111, 'Delivered'),

    (42, 2, 112, 'Pending'),

    (43, 3, 113, 'Delivered'),

    (44, 4, 114, 'Processing'),

    (45, 5, 111, 'Delivered');

select \* from foodweb.Delivery;

3. use foodweb;

CREATE TABLE foodweb.Driver (

    driver\_id INT PRIMARY KEY,

    name VARCHAR(255),

    phone\_number VARCHAR(20),

    vehicle\_type VARCHAR(50)

);

insert into driver values(111,"Suresh","268418461","Suzuzki");

insert into driver values(112,"naresh","285618461","Pulsar");

insert into driver values(113,"qureshi","414461","Toyoto");

insert into driver values(114,"ansh","268448178461","bullet");

select \* from foodweb.driver;

4. use foodweb;

CREATE TABLE foodweb.Menu (

menu\_id INT PRIMARY KEY,

restaurant\_id INT,

name VARCHAR(255),

description TEXT,

price DECIMAL(10,2),

FOREIGN KEY (restaurant\_id) REFERENCES Restaurant(restaurant\_id)

);

INSERT INTO foodweb.Menu (menu\_id, restaurant\_id, name, description, price)

VALUES

(1, 511, 'Item 1', 'Description 1', 10.99),

(2, 512, 'Item 2', 'Description 2', 8.99),

(3, 513, 'Item 3', 'Description 3', 12.50),

(4, 514, 'Item 4', 'Description 4', 15.99),

(5, 515, 'Item 5', 'Description 5', 9.99),

(6, 514, 'Item 6', 'Description 6', 11.99),

(7, 513, 'Item 7', 'Description 7', 14.99),

(8, 515, 'Item 8', 'Description 8', 16.50),

(9, 511, 'Item 9', 'Description 9', 13.99),

(10,514, 'Item 10', 'Description 10', 10.49);

select \* from foodweb.Menu;

alter table foodweb.Menu drop column menu\_id;

5. use foodweb;

CREATE TABLE foodweb.order (

    order\_id INT PRIMARY KEY,

    customer\_id INT,

    restaurant\_id INT,

    delivery\_address VARCHAR(255),

    total\_price DECIMAL(10,2),

    status VARCHAR(50),

    FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

    FOREIGN KEY (restaurant\_id) REFERENCES Restaurant(restaurant\_id)

);

INSERT INTO foodweb.order (order\_id, customer\_id, restaurant\_id, delivery\_address, total\_price, status)

VALUES

    (1, 1, 511, 'Delivery Address 1', 25.99, 'Pending'),

    (2, 2, 512, 'Delivery Address 2', 18.99, 'Delivered'),

    (3, 3, 513, 'Delivery Address 3', 35.50, 'Pending'),

    (4, 4, 514, 'Delivery Address 4', 45.00, 'Processing'),

    (5, 5, 515, 'Delivery Address 5', 29.99, 'Delivered');

select \* from foodweb.order;

6. use foodweb;

CREATE TABLE foodweb.Payment (

    payment\_id INT PRIMARY KEY,

    order\_id INT,

    amount DECIMAL(10,2),

    payment\_method VARCHAR(50),

    FOREIGN KEY (order\_id) REFERENCES foodweb.order(order\_id)

);

INSERT INTO foodweb.Payment (payment\_id, order\_id, amount, payment\_method)

VALUES

    (1, 1, 25.99, 'Cash'),

    (2, 2, 18.99, 'Credit Card'),

    (3, 3, 35.50, 'Cash'),

    (4, 4, 45.00, 'Credit Card'),

    (5, 5, 29.99, 'Cash');

select \* from foodweb.Payment;

7. use foodweb;

CREATE TABLE foodweb.Promotion (

    promotion\_id INT PRIMARY KEY,

    restaurant\_id INT,

    promotion\_details TEXT,

    start\_date DATE,

    end\_date DATE,

    FOREIGN KEY (restaurant\_id) REFERENCES foodweb.restaurant(restaurant\_id)

);

INSERT INTO foodweb.Promotion VALUES (1, 511, '20% off on all orders', '2024-09-01', '2024-09-10');

INSERT INTO foodweb.Promotion VALUES (2, 512, 'Buy 1 Get 1 Free on pizzas', '2024-09-05', '2024-09-15');

INSERT INTO foodweb.Promotion VALUES (3, 513, 'Free dessert with every meal', '2024-09-10', '2024-09-20');

INSERT INTO foodweb.Promotion VALUES (4, 514, '10% off on all vegetarian dishes', '2024-09-12', '2024-09-22');

select \* from foodweb.promotion;

8. use foodweb;

CREATE TABLE foodweb.Rating (

    rating\_id INT PRIMARY KEY,

    order\_id INT,

    stars INT,

    comment TEXT,

    FOREIGN KEY (order\_id) REFERENCES foodweb.order(order\_id)

);

INSERT INTO foodweb.Rating(rating\_id,order\_id,stars,comment)

VALUES

(120,1,4,'good');

INSERT INTO foodweb.Rating(rating\_id,order\_id,stars,comment)

VALUES

(121,2,3,'Not so bad');

INSERT INTO foodweb.Rating(rating\_id,order\_id,stars,comment)

VALUES

(122,3,5,'very nice');

INSERT INTO foodweb.Rating(rating\_id,order\_id,stars,comment)

VALUES

(123,4,3,'Average');

INSERT INTO foodweb.Rating(rating\_id,order\_id,stars,comment)

VALUES

(124,5,5,'fabulous');

select \* from Rating;

9. CREATE TABLE `restaurant` (

  `restaurant\_id` int NOT NULL,

  `restaurant\_name` varchar(255) DEFAULT NULL,

  `location` varchar(255) DEFAULT NULL,

  `cuisine\_type` varchar(255) DEFAULT NULL,

  `contact\_info` varchar(255) DEFAULT NULL,

  `rating` float DEFAULT NULL,

  PRIMARY KEY (`restaurant\_id`)

) ;

INSERT INTO `restaurant` VALUES (511,'NextDoorRestaurant','LB nagar','Chicken tikka','+12-23898392',4.8),

(512,'Mr.David Restro and bar','Jubilee-hills','Chicken-Sandwich','+12-64646844',4.4),

(513,'Non-vegeterian restaurant','hayathnagar','Fry Fish','+12-23898392',4.1),

(514,'Mehfil','dilsukhnagar','Chicken shawarma','+12-65484648',3.8),

(515,'Bistro','Hitech city','cocktails & pizza','+12-68478482',4.88),

(516,'Dominos','Kukatapally','Pizza','+12-284184148',4.8);

Select \* from restaurant;

Conclusion:

The Online Food Delivery System offers a convenient platform for customers to order food from various restaurants, making the food delivery process faster and more efficient. By integrating features such as user authentication, real-time order tracking, secure payment options, and feedback mechanisms, the system enhances the overall customer experience. Restaurants can easily manage their menus and orders, while delivery personnel can efficiently track and fulfill deliveries. The system also ensures that administrators can monitor and manage the platform smoothly. This project demonstrates how technology can simplify daily tasks and improve service quality in the food delivery industry.