Faculty Of Computers & Artificial Intelligence

University of Sadat City

Online Food Order and Delivery

Subject	Database Project
Team Name	Problematic
Level	Three
Department	Information Systems (IS)
Supervisor	Dr/ Ahmed Gamal & Eng/ David Kamal

Team Names:

01	Ahmed Abdullah (TL)	
02	Mohamed Ashraf	
03	Salma Yasser	
04	Ola Farhat	

Business Use of an Online Food Ordering and Delivery System

1. Problem Statement (Scenario)

The foodservice industry has seen a growing need for digital solutions that streamline the process of ordering and delivering food, particularly with the increasing popularity of takeout and home delivery. Customers expect convenience, efficiency, and multiple options when it comes to ordering food. Traditional methods, such as calling the restaurant directly, are often inefficient, especially during peak hours, and do not provide transparency regarding delivery status or food options.

2. Business Need

An **Online Food Ordering and Delivery System** serves as a solution to:

1. **Enhance Customer Experience**: Provide a convenient platform for users to browse restaurant menus, customize orders, and receive timely deliveries. The system offers a seamless user experience, making the ordering process faster and more transparent.
2. **Optimize Restaurant Operations**: Restaurants can manage incoming orders in an organized manner, reducing the chances of miscommunication or errors. This system helps restaurants optimize their workflow, especially during busy times.
3. **Streamline Delivery Services**: The system helps delivery agents receive clear instructions, manage multiple deliveries, and track their assigned tasks effectively. It provides real-time updates to customers, enhancing overall satisfaction.
4. **Data Collection for Business Insights**: By maintaining customer, order, and delivery information in a well-structured database, the system enables data-driven decision-making. Restaurants can analyze ordering patterns, popular menu items, peak hours, and more.

Key Business Use Cases

1. **Centralized Ordering Platform**
    - The platform lets customers explore multiple restaurants in one place, with options to filter by cuisine and dietary preferences. They can place orders from different restaurants without switching between apps or websites.
        - **For Customers:** They can view various options easily without making calls or visiting multiple sites.
        - **For Restaurants:** They reach a wider audience by being part of a collective platform.
2. **Real-Time Order and Delivery Updates**
    - Once an order is placed, customers receive updates at each stage—when food preparation starts, when the order is ready, and when it's on the way.
        - **For Customers:** This tracking feature reduces anxiety, provides clear expectations on delivery time, and boosts satisfaction.
3. **Enhanced Restaurant Efficiency**
    - Restaurants receive organized orders through the system, reducing errors from phone orders. This setup helps them better manage food preparation and resources, especially for incoming orders.
4. **Improved Customer Relationship Management (CRM)**

- Customer data, including past orders and preferences, helps build personalized experiences. The system can:
  - Suggest favorite dishes.
  - Offer loyalty rewards and promotions.
  - Gather feedback for customer satisfaction.

5. **Delivery Management**
   - Delivery agents are assigned tasks with clear details: pickup location, drop-off address, and customer information.
     - **For Delivery Agents:** Route optimization ensures faster deliveries and greater efficiency in daily tasks.

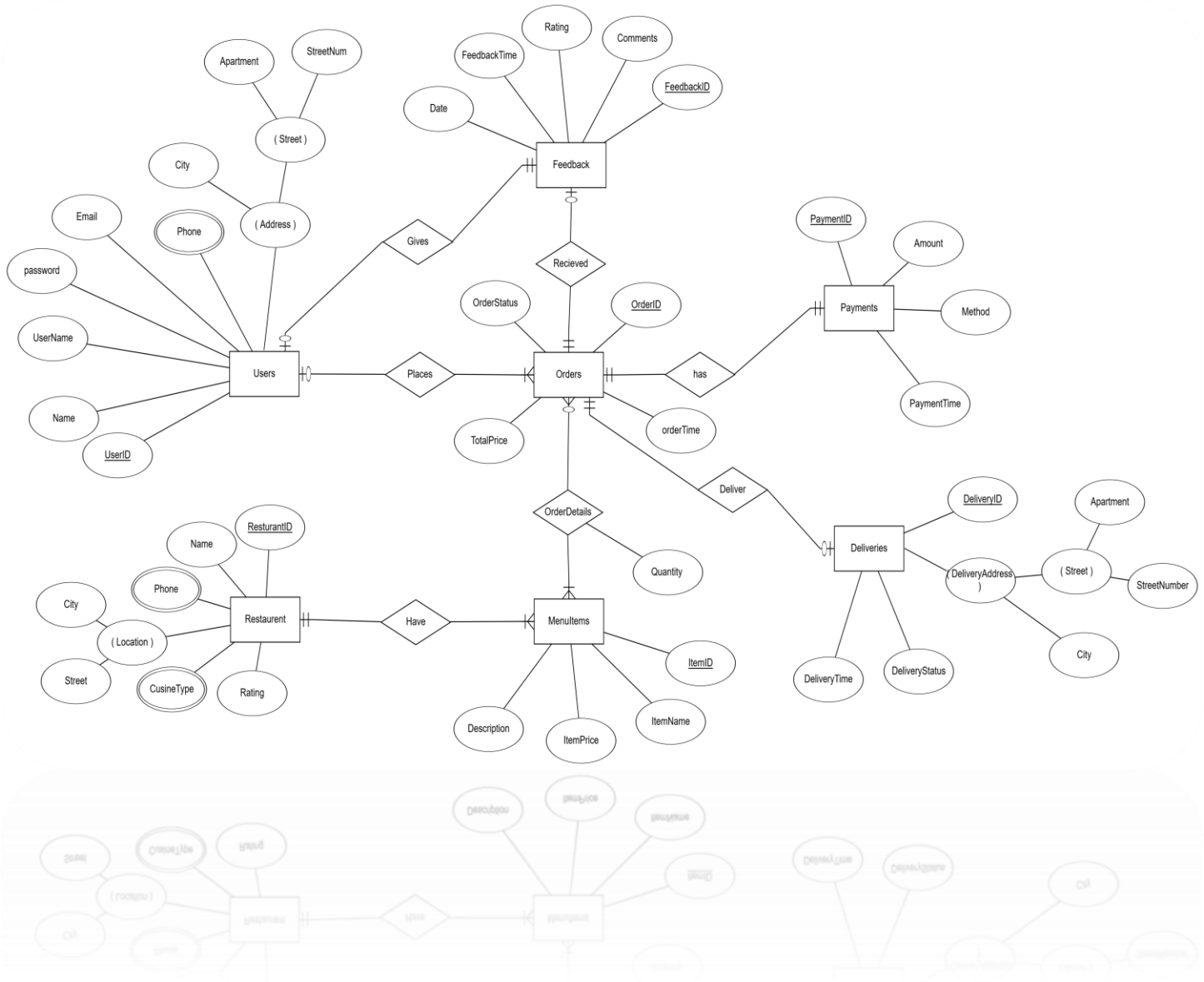
Value Proposition

- **For Restaurants:**
  - **Boosted Sales:** By reaching more customers through an online presence.
  - **Operational Efficiency:** Structured order processing means quicker order fulfillment and fewer errors.
  - **Insightful Data:** Restaurants can use sales data to improve menu choices and plan for busy times.
- **For Customers:**
  - **Convenience:** Easy browsing, ordering, and payment in one app.
  - **Time-Saving:** Avoids wait times and busy phone lines.
  - **Informed Choices:** Menus include descriptions, reviews, and ratings for better decision-making.
- **For Delivery Agents:**
  - **Efficient Management:** Clear tasks, optimized routes, and fewer delays improve job satisfaction.
  - **Reduced Idle Time:** Well-organized deliveries mean less downtime between orders.

Expected Outcomes

- **Higher Customer Retention:** A user-friendly system, loyalty programs, and order transparency encourage repeat business.
- **Shorter Order Handling Time:** No phone orders are needed, and orders can be managed efficiently.
- **Increased Revenue:** Restaurants serve more customers, especially during peak times, and reach new audiences.
- **Targeted Marketing:** Customer preferences drive specific promotions, increasing order frequency

ER Diagram



System Overview

This ERD represents a comprehensive food ordering and delivery system that includes:

User management with secure authentication and detailed address storage.

- Feedback collection for orders to ensure customer satisfaction.
- Order processing with integrated payment and delivery systems.
- Restaurant and menu management to maintain flexibility in item offerings.
-

Analysis of the Entity-Relationship Diagram (ERD)

The ERD provides a structured representation of the entities, attributes, and relationships involved in the system. Below is a detailed analysis of each entity, its attributes, and its relationships:

---

1. Users

- **Attributes**:
  - `UserID` (Primary Key): Unique identifier for each user.
  - `Name`: Full name of the user.
  - `UserName`: A unique username chosen by the user.
  - `Password`: Secure password for account authentication.
  - `Email`: Email address of the user.
  - `Phone`: Contact number of the user.
  - `Address` (Composite Attribute):
    - `Street`: Street name.
    - `StreetNum`: Street number.
    - `Apartment`: Apartment or unit number.
    - `City`: City of residence.
- **Relationships**:
  - **Places**: A user places an order (1-to-many relationship with `Orders`).
  - **Gives**: A user can provide feedback (1-to-many relationship with `Feedback`).

---

2. Feedback

- **Attributes**:
  - `FeedbackID` (Primary Key): Unique identifier for each feedback entry.
  - `Rating`: Numerical or categorical rating given by the user.
  - `Comments`: Textual comments from the user.
  - `Date`: Date the feedback was submitted.
  - `FeedbackTime`: Time the feedback was submitted.
- **Relationships**:
  - **Gives**: Feedback is given by a user (1-to-many relationship with `Users`).
  - **Received**: Feedback is associated with a specific order (1-to-1 relationship with `Orders`).

---

3. Orders

- **Attributes**:
  - `OrderID` (Primary Key): Unique identifier for each order.
  - `OrderStatus`: Status of the order (e.g., Pending, Completed).
  - `TotalPrice`: Total price of the order.
  - `OrderTime`: Time the order was placed.
  - `OrderDetails`: Description or summary of the order.
- **Relationships**:
  - **Places**: Orders are placed by users (many-to-1 relationship with `Users`).
  - **Has**: An order is associated with a payment (1-to-1 relationship with `Payments`).
  - **Deliver**: An order is associated with a delivery (1-to-1 relationship with `Deliveries`).
  - **OrderDetails**: An order contains specific menu items (1-to-many relationship with `MenuItems`).

---

4. Payments

- **Attributes**:
  - `PaymentID` (Primary Key): Unique identifier for each payment.
  - `Amount`: The amount paid for the order.
  - `Method`: Payment method used (e.g., Credit Card, Cash).
  - `PaymentTime`: Time the payment was processed.
- **Relationships**:
  - **Has**: A payment is associated with an order (1-to-1 relationship with `Orders`).

---

5. Deliveries

- **Attributes**:
  - `DeliveryID` (Primary Key): Unique identifier for each delivery.
  - `DeliveryAddress` (Composite Attribute):
    - `Street`: Street name.
    - `StreetNumber`: Street number.
    - `Apartment`: Apartment or unit number.
    - `City`: City of delivery.
  - `DeliveryTime`: Time the delivery was completed.
  - `DeliveryStatus`: Status of the delivery (e.g., Pending, Completed).
- **Relationships**:
  - **Deliver**: A delivery is associated with an order (1-to-1 relationship with `Orders`).

---

6. Restaurant

- **Attributes**:
  - `RestaurantID` (Primary Key): Unique identifier for each restaurant.
  - `Name`: Name of the restaurant.
  - `Phone`: Contact number of the restaurant.
  - `Location` (Composite Attribute):
    - `Street`: Street name.
    - `City`: City where the restaurant is located.
  - `CuisineType`: Type of cuisine offered by the restaurant.
  - `Rating`: Overall rating of the restaurant.
- **Relationships**:
  - **Have**: Restaurants have menu items (1-to-many relationship with `MenuItems`).

---

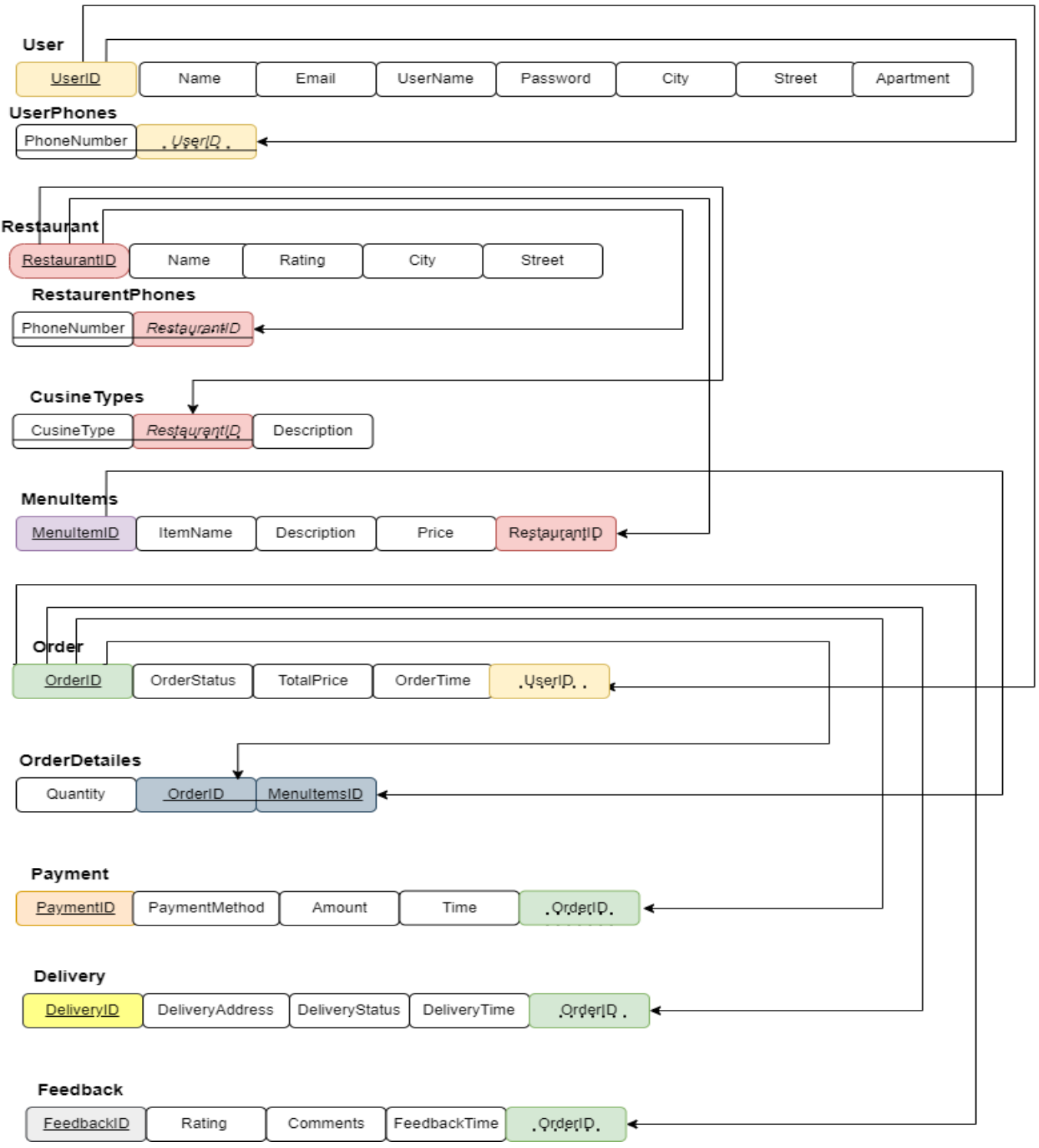
7. MenuItems

- **Attributes**:
  - `ItemID` (Primary Key): Unique identifier for each menu item.
  - `ItemName`: Name of the menu item.
  - `Description`: Description of the menu item.
  - `ItemPrice`: Price of the menu item.
- **Relationships**:
  - **Have**: Menu items belong to restaurants (many-to-1 relationship with `Restaurant`).
  - **OrderDetails**: Menu items are part of an order (many-to-many relationship with `Orders`).

Key Relationships

1. **User-Orders Relationship**: A user places multiple orders, creating a 1-to-many relationship between `Users` and `Orders`.
2. **Orders-Payments Relationship**: Each order has a single associated payment, forming a 1-to-1 relationship.
3. **Orders-Deliveries Relationship**: Each order is associated with one delivery, forming another 1-to-1 relationship.
4. **Orders-MenuItems Relationship**: An order can contain multiple menu items, forming a many-to-many relationship.
5. **Restaurant-MenuItems Relationship**: Restaurants have multiple menu items, creating a 1-to-many relationship.

Mapping Process



Analysis of the Mapping Process in the ERD

The mapping process shown in the diagram illustrates the normalization of the data model and the creation of tables and relationships for relational database implementation. Below is a breakdown of the mapping process:

Key Highlights of the Mapping Process

1. User

- The `User` entity is mapped to a table with attributes: `UserID`, `Name`, `Email`, `UserName`, `Password`, `City`, `Street`, and `Apartment`.
- **Normalization**:
  - A separate table, `UserPhones`, is created for the `Phone` attribute, allowing a user to have multiple phone numbers.
  - The `UserPhones` table contains:
    - `PhoneNumber`: Primary key.
    - `UserID`: Foreign key linking to the `User` table.

---

2. Restaurant

- The `Restaurant` entity is mapped to a table with attributes: `RestaurantID`, `Name`, `Rating`, `City`, and `Street`.
- **Normalization**:
  - A separate table, `RestaurantPhones`, is created for the `Phone` attribute, allowing a restaurant to have multiple phone numbers.
  - The `RestaurantPhones` table contains:
    - `PhoneNumber`: Primary key.
    - `RestaurantID`: Foreign key linking to the `Restaurant` table.
  - A separate table, `CuisineTypes`, is created to store cuisine information for each restaurant:
    - `CuisineType`: Name of the cuisine.
    - `RestaurantID`: Foreign key linking to the `Restaurant` table.
    - `Description`: Optional description of the cuisine type.

---

3. MenuItems

- The `MenuItems` entity is mapped to a table with attributes:
  - `MenuItemID`: Primary key.
  - `ItemName`: Name of the menu item.
  - `Description`: Description of the menu item.
  - `Price`: Price of the menu item.
  - `RestaurantID`: Foreign key linking to the `Restaurant` table.

---

4. Order

- The `Order` entity is mapped to a table with attributes:
  - `OrderID`: Primary key.
  - `OrderStatus`: Status of the order (e.g., Pending, Completed).
  - `TotalPrice`: Total price of the order.
  - `OrderTime`: Timestamp when the order was placed.
  - `UserID`: Foreign key linking to the `User` table.

---

5. OrderDetails

- A new table, `OrderDetails`, is created to handle the many-to-many relationship between `Order` and `MenuItems`.
- The `OrderDetails` table contains:
  - `OrderID`: Foreign key linking to the `Order` table.
  - `MenuItemID`: Foreign key linking to the `MenuItems` table.
  - `Quantity`: Quantity of the specific menu item in the order.

---

6. Payment

- The `Payment` entity is mapped to a table with attributes:
  - `PaymentID`: Primary key.
  - `PaymentMethod`: Method of payment (e.g., Cash, Credit Card).
  - `Amount`: Amount paid for the order.
  - `Time`: Timestamp of the payment.
  - `OrderID`: Foreign key linking to the `Order` table.

---

7. Delivery

- The `Delivery` entity is mapped to a table with attributes:
  - `DeliveryID`: Primary key.
  - `DeliveryAddress`: Address of the delivery.
  - `DeliveryStatus`: Status of the delivery (e.g., Pending, Delivered).
  - `DeliveryTime`: Timestamp of the delivery.
  - `OrderID`: Foreign key linking to the `Order` table.

---

8. Feedback

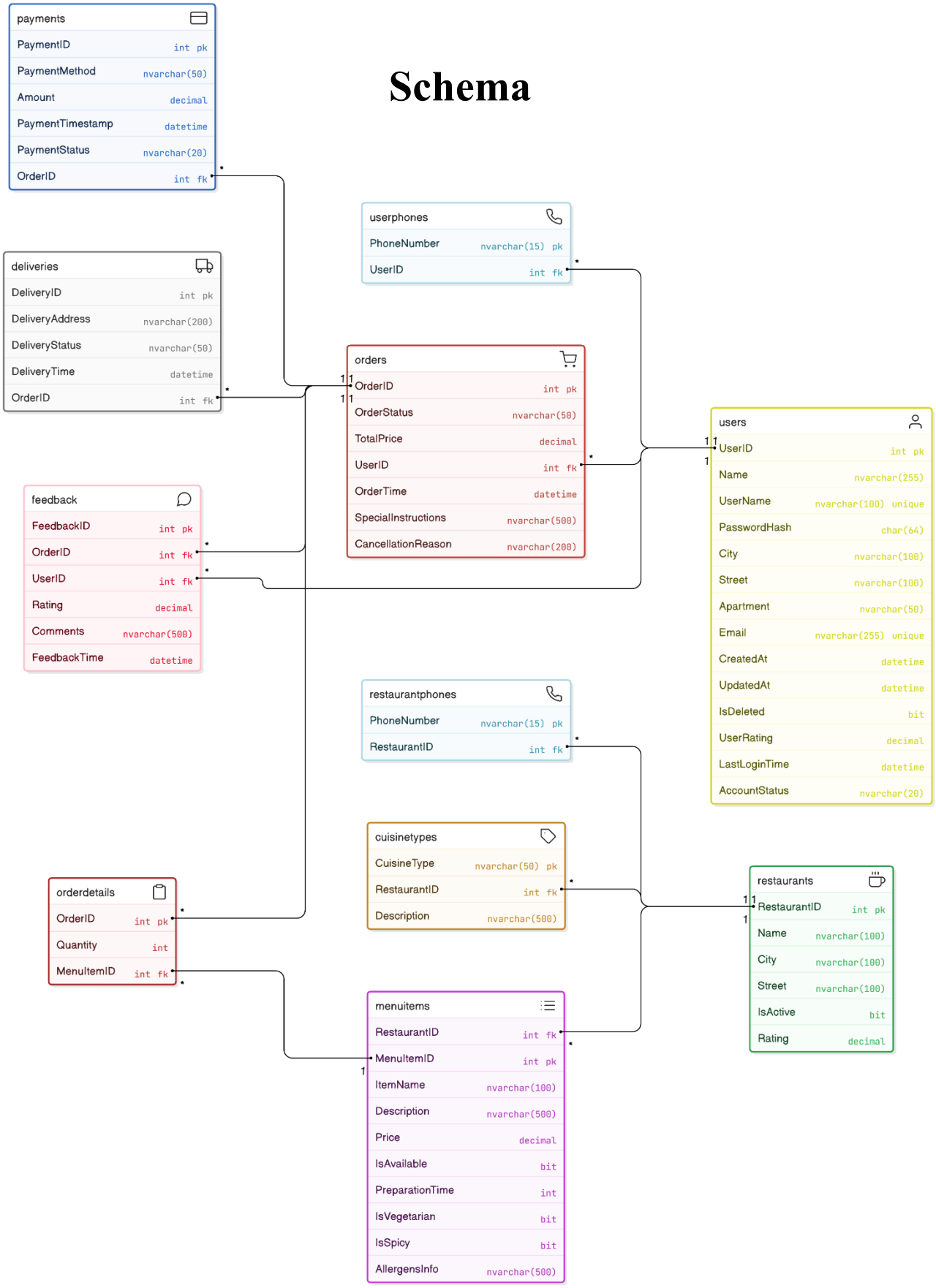
- The `Feedback` entity is mapped to a table with attributes:
  - `FeedbackID`: Primary key.
  - `Rating`: Rating provided by the user.
  - `Comments`: Textual feedback provided by the user.
  - `FeedbackTime`: Timestamp of when the feedback was submitted.
  - `OrderID`: Foreign key linking to the `Order` table.

Summary of Normalization Process

1. **Entity Splitting**:
   - Composite attributes (`Phone` for `User` and `Restaurant`, `CuisineType` for `Restaurant`) were normalized into separate tables.
2. **Many-to-Many Relationships**:
   - `OrderDetails` was created to handle the many-to-many relationship between `Orders` and `MenuItems`.
3. **Foreign Key Constraints**:
   - Relationships between entities were converted into foreign key constraints to maintain referential integrity (e.g., `UserID` in `Order`, `OrderID` in `Payment` and `Delivery`).

This mapping process ensures that the data model is normalized, avoids redundancy, and is optimized for implementation in a relational database.

Schema


payments
PaymentID int pk
PaymentMethod nvarchar(50)
Amount decimal
PaymentTimestamp datetime
PaymentStatus nvarchar(20)
OrderID int fk
userphones
PhoneNumber nvarchar(15) pk
UserID int fk
deliveries
DeliveryID int pk
DeliveryAddress nvarchar(200)
DeliveryStatus nvarchar(50)
DeliveryTime datetime
OrderID int fk
orders
OrderID int pk
OrderStatus nvarchar(50)
TotalPrice decimal
UserID int fk
OrderTime datetime
SpecialInstructions nvarchar(500)
CancellationReason nvarchar(200)
users
UserID int pk
Name nvarchar(255)
UserName nvarchar(100) unique
PasswordHash char(64)
City nvarchar(100)
Street nvarchar(100)
Apartment nvarchar(50)
Email nvarchar(255) unique
CreatedAt datetime
UpdatedAt datetime
IsDeleted bit
UserRating decimal
LastLoginTime datetime
AccountStatus nvarchar(20)
feedback
FeedbackID int pk
OrderID int fk
UserID int fk
Rating decimal
Comments nvarchar(500)
FeedbackTime datetime
restaurantphones
PhoneNumber nvarchar(15) pk
RestaurantID int fk
cuisinetypes
CuisineType nvarchar(50) pk
RestaurantID int fk
Description nvarchar(500)
restaurants
RestaurantID int pk
Name nvarchar(100)
City nvarchar(100)
Street nvarchar(100)
IsActive bit
Rating decimal
orderdetails
OrderID int pk
Quantity int
MenuItemID int fk
menuitems
RestaurantID int fk
MenuItemID int pk
ItemName nvarchar(100)
Description nvarchar(500)
Price decimal
IsAvailable bit
PreparationTime int
IsVegetarian bit
IsSpicy bit
AllergensInfo nvarchar(500)

Analysis of the Enhanced Schema

The enhanced schema illustrates a more detailed and refined version of the relational database design for the online order delivery system. It builds upon the ERD and the mapping process, introducing additional attributes, normalization, and design considerations. Below is an analysis of the changes and additions:

Key Additions and Enhancements

1. Users

- **Attributes Added**:
  - `PasswordHash`: Replaces plain `Password` for secure storage of hashed passwords.
  - `CreatedAt` and `UpdatedAt`: Track the creation and modification timestamps of user records.
  - `IsDeleted`: Indicates if the user account is soft-deleted (logical deletion).
  - `UserRating`: A rating metric for the user (e.g., aggregated feedback score).
  - `LastLoginTime`: Timestamp of the user's last login for activity tracking.
  - `AccountStatus`: Tracks the status of the account (e.g., Active, Suspended).
- **Relationships**:
  - Remains the same with `UserPhones` and `Orders`.

---

2. UserPhones

- **Attributes**:
  - `PhoneNumber`: Primary key.
  - `UserID`: Foreign key linking to `Users`.
- **Enhancements**:
  - Additional normalization for managing multiple phone numbers for users.

---

3. Restaurants

- **Attributes Added**:
  - `IsActive`: Indicates if the restaurant is actively available for taking orders.
- **Relationships**:
  - Unchanged, still connected to `RestaurantPhones`, `CuisineTypes`, and `MenuItems`.

---

4. **RestaurantPhones**
   Attributes:

   - `PhoneNumber`: Primary key.
   - `RestaurantID`: Foreign key linking to `Restaurants`.

- **Enhancements**:
  - No significant change.

---

5. CuisineTypes

- **Attributes**:
  - `CuisineType`: Primary key.
  - `RestaurantID`: Foreign key linking to `Restaurants`.
  - `Description`: Description of the cuisine.
- **Enhancements**:
  - No significant change, but essential for supporting diverse cuisine types.

---

6. MenuItems

- **Attributes Added**:
  - `IsAvailable`: Indicates if the menu item is available for ordering.
  - `PreparationTime`: Estimated time to prepare the menu item (in minutes).
  - `IsVegetarian`: Boolean flag for vegetarian items.
  - `IsSpicy`: Boolean flag for spicy items.
  - `AllergensInfo`: Contains information about allergens in the menu item.
- **Relationships**:
  - Links to `Restaurants` (unchanged).
  - Forms a many-to-many relationship with `Orders` via `OrderDetails`.

---

7. Orders

- **Attributes Added**:
  - `SpecialInstructions`: Allows users to specify additional instructions for their orders.
  - `CancellationReason`: Tracks reasons if an order is canceled.
- **Relationships**:
  - Unchanged, linked to `Users`, `OrderDetails`, `Feedback`, `Payments`, and `Deliveries`.

---

8. OrderDetails

- **Attributes**:
  - `OrderID`: Foreign key linking to `Orders`.
  - `MenuItemID`: Foreign key linking to `MenuItems`.
  - `Quantity`: Quantity of the specific menu item in the order.
- **Enhancements**:
  - Provides more granularity for tracking individual menu items within orders.

---

9. Payments

- **Attributes Added**:
  - `PaymentStatus`: Tracks the status of the payment (e.g., Successful, Failed, Pending).
- **Relationships**:
  - Unchanged, still linked to `Orders`.

---

10. Deliveries

- **Attributes Added**:
  - `DeliveryTime`: Tracks the exact time of delivery.
- **Relationships**:
  - Unchanged, linked to `Orders`.

---

11. Feedback

- **Attributes Added**:
  - No major additions, but structure remains robust.
- **Relationships**:
  - Linked to both `Orders` and `Users`.

Summary of Schema Enhancements

1. **Security and Tracking**:
   - Passwords are now securely stored using hashing (`PasswordHash`).
   - Audit fields like `CreatedAt`, `UpdatedAt`, and `LastLoginTime` are added for tracking user and system activity.
2. **Flexibility**:
   - `SpecialInstructions` and `CancellationReason` fields in `Orders` enhance user customization and improve record-keeping.
3. **Product Information**:
   - Menu items are more descriptive with fields like `PreparationTime`, `IsVegetarian`, `IsSpicy`, and `AllergensInfo`.
4. **Operational Support**:
   - `IsDeleted` and `AccountStatus` provide better user and account management capabilities.
5. **Reliability**:
   - Fields like `PaymentStatus` and `DeliveryTime` improve the accuracy and reliability of payment and delivery tracking.

Implementation Phase

DDL(Defining the Tables and Constraints)

Below is the detailed explanation and documentation for the SQL Data Definition Language (DDL) commands used to create the tables in the Online Order Delivery System database:

---

1. Users Table

- **Purpose**: Stores user information, including login credentials and personal details.
- **Key Constraints**:
  - UserID: Primary key, auto-incremented.
  - UserName and Email: Unique constraints to prevent duplicates.
  - PasswordHash: Ensures secure storage of passwords.
  - UserRating: Checked to ensure it remains between 1 and 5.
  - AccountStatus: Restricted to valid states ('Active', 'Suspended', 'Blocked').
  - IsDeleted: Soft deletion flag for user accounts.
- **Special Constraints**:
  - CHECK constraint on Email for email format validation.

---

2. UserPhones Table

- **Purpose**: Allows users to associate multiple phone numbers with their account.
- **Key Constraints**:
  - Composite primary key: PhoneNumber and UserID.
  - Foreign key: UserID references Users(UserID) with cascading deletion.
  - PhoneNumber format is validated using a CHECK constraint.

---

3. Restaurants Table

- **Purpose**: Stores restaurant details, including location and rating.
- **Key Constraints**:
  - RestaurantID: Primary key, auto-incremented.
  - Rating: Ensures values are between 0 and 5.
  - IsActive: Default value is 1 to indicate active restaurants.

---

4. RestaurantPhones Table

- **Purpose**: Supports multiple phone numbers for each restaurant.
- **Key Constraints**:

- Composite primary key: PhoneNumber and RestaurantID.
- Foreign key: RestaurantID references Restaurants(RestaurantID) with cascading deletion.
- PhoneNumber format is validated using a CHECK constraint.

---

5. CuisineTypes Table

- **Purpose**: Defines the types of cuisines offered by each restaurant.
- **Key Constraints**:
  - Composite primary key: CuisineType and RestaurantID.
  - Foreign key: RestaurantID references Restaurants(RestaurantID) with cascading deletion.

---

6. MenuItems Table

- **Purpose**: Stores information about menu items provided by restaurants.
- **Key Constraints**:
  - MenuItemID: Primary key, auto-incremented.
  - Price: Must be greater than 0.
  - Foreign key: RestaurantID references Restaurants(RestaurantID) with cascading deletion.
- **Additional Attributes**:
  - IsAvailable: Indicates item availability (default is 1).
  - IsVegetarian, IsSpicy: Flags to categorize menu items.
  - AllergensInfo: Text field for allergen details.

---

7. Orders Table

- **Purpose**: Tracks customer orders, including status and special instructions.
- **Key Constraints**:
  - OrderID: Primary key, auto-incremented.
  - OrderStatus: Restricted to valid states ('Pending', 'Completed', 'Cancelled').
  - TotalPrice: Must be greater than or equal to 0.
  - Foreign key: UserID references Users(UserID) with cascading deletion.
- **Additional Attributes**:
  - SpecialInstructions: Default text for unspecified instructions.
  - CancellationReason: Default text for orders not canceled.

---

8. OrderDetails Table

- **Purpose**: Defines the many-to-many relationship between orders and menu items.
- **Key Constraints**:
  - Composite primary key: OrderID and MenuItemID.
  - Foreign keys:

        - OrderID references Orders(OrderID) with cascading deletion.
        - MenuItemID references MenuItems(MenuItemID) with cascading deletion.
    - Quantity: Must be greater than 0.

---

9. Payments Table

- **Purpose**: Tracks payment details for orders.
- **Key Constraints**:
    - PaymentID: Primary key, auto-incremented.
    - PaymentMethod: Restricted to valid types ('Cash', 'Online').
    - Amount: Must be greater than 0.
    - Foreign key: OrderID references Orders(OrderID) with cascading deletion.
- **Additional Attributes**:
    - PaymentStatus: Default value is Pending with valid states ('Pending', 'Processing', 'Completed', 'Failed', 'Refunded').

---

10. Deliveries Table

- **Purpose**: Tracks delivery details for orders.
- **Key Constraints**:
    - DeliveryID: Primary key, auto-incremented.
    - DeliveryStatus: Restricted to valid states ('Pending', 'In Transit', 'Delivered').
    - Foreign key: OrderID references Orders(OrderID) with cascading deletion.

---

11. Feedback Table

- **Purpose**: Stores user feedback for completed orders.
- **Key Constraints**:
    - FeedbackID: Primary key, auto-incremented.
    - Rating: Must be between 1 and 5.
    - Foreign keys:
        - OrderID references Orders(OrderID) with cascading deletion.
        - UserID references Users(UserID) with cascading deletion.

Summary of Constraints

1. **Primary Keys**: Ensure unique identification of each record in all tables.
2. **Foreign Keys**: Establish relationships between tables and enforce referential integrity.
3. **Check Constraints**: Validate data formats, ranges, and predefined values.
4. **Default Values**: Ensure consistent behavior for optional attributes.
5. **Cascading Deletions**: Automatically remove dependent records when parent records are deleted.

Implementation Phase:

Data Query Language (DQL) and Data Manipulation Language (DML)

The provided SQL script includes data insertion, updating, deletion, and creation of views. Here's the detailed documentation of the implementation phase:

---

1. Data Insertion

The INSERT statements populate each table with sample data, demonstrating how the system initializes its database with users, restaurants, menu items, orders, payments, deliveries, and feedback.

---

2. Data Updates

The UPDATE statements modify existing records to demonstrate changes in the database.

Highlights of Updates:

1. **Users**:
   - Updated Password, Street, and Apartment for UserID 1.
2. **UserPhones**:
   - Modified phone number for UserID 2.
3. **Restaurants**:
   - Changed the street address for RestaurantID 1.
4. **RestaurantPhones**:
   - Updated phone number for RestaurantID 2.
5. **MenuItems**:
   - Adjusted the price for MenuItemID 3.
6. **Orders**:
   - Changed OrderStatus to Completed and updated the TotalPrice.
7. **OrderDetails**:
   - Increased Quantity for a specific order and menu item combination.
8. **Payments**:
   - Adjusted the payment amount for PaymentID 2.
9. **Deliveries**:
   - Changed the DeliveryStatus for DeliveryID 1 to Delivered.

10. **Feedback**:
    - Updated the Rating and Comments for FeedbackID 3.

---

3. Data Deletion

The DELETE statements remove specific records from the database to simulate data lifecycle management.

Highlights of Deletions:

1. **Users**:
    - Deleted the user with UserID 3.
2. **UserPhones**:
    - Removed the phone number associated with UserID 1.
3. **Restaurants**:
    - Deleted RestaurantID 2 and associated data.
4. **RestaurantPhones**:
    - Removed the phone number for RestaurantID 3.
5. **MenuItems**:
    - Deleted MenuItemID 2.
6. **Orders**:
    - Removed an unused order with OrderID 4.
7. **OrderDetails**:
    - Deleted the link between OrderID 1 and MenuItemID 1.
8. **Payments**:
    - Deleted PaymentID 3.
9. **Deliveries**:
    - Removed the delivery with DeliveryID 2.
10. **Feedback**:
    - Deleted FeedbackID 1.

---

4. Views

The CREATE VIEW statements define simplified ways to query the database for specific information.

Highlights of Views:

1. **View_Users**:
   - Displays basic user information: UserID, UserName, Password, City, Street, Apartment, and Role.
2. **View_UserPhones**:
   - Lists phone numbers associated with users.
3. **View_Restaurants**:
   - Shows restaurant details: RestaurantID, Name, City, and Street.
4. **View_RestaurantPhones**:
   - Lists phone numbers for each restaurant.
5. **View_CuisineTypes**:
   - Displays cuisine types and their associated restaurants.
6. **View_MenuItems**:
   - Lists menu items with their attributes.
7. **View_Orders**:
   - Displays order information: OrderID, OrderStatus, TotalPrice, and UserID.
8. **View_OrderDetails**:
   - Shows the details of orders, including OrderID, MenuItemID, and Quantity.
9. **View_Payments**:
   - Displays payment details: PaymentID, PaymentMethod, Amount, and OrderID.
10. **View_Deliveries**:
    - Lists delivery details, including DeliveryID, DeliveryAddress, DeliveryStatus, and OrderID.
11. **View_Feedback**:
    - Displays feedback, including FeedbackID, Rating, Comments, and OrderID.

Summary of DQL and DML Operations

- **Insertion**: Populated tables with initial data to represent real-world scenarios.
- **Updates**: Demonstrated the ability to modify specific records based on conditions.
- **Deletions**: Ensured proper removal of records and cascade deletions where necessary.
- **Views**: Simplified querying by creating logical representations of the data.