1. **Delivery Form** (`Delivery` class):

    - **Fields**: `number` and `food` store customer information.
    - **Constructor**: Initializes the form with the provided `number` and `food`.
    - **btnOrder_Click Event Handler**:
        - **Database Connection**: Uses a connection string to connect to the SQL Server database.
        - **Retrieve Food Details**: Fetches `food_id` and `price` from the `food` table.
        - **Retrieve Order ID**: Gets the next available `order_id` from the `order_customer` table.
        - **Insert Order**: Inserts a new order into the `order_customer` table.
        - **Insert Delivery Reservation**: Inserts delivery details into the `delivary` table.
        - **Success Notification**: Displays a success message and opens a `Success` form.

2. **Admin Form** (`Admin` class):

    - **Constructor**: Initializes the admin form.
    - This form currently does not contain any additional logic or event handlers.

3. **CustomerChoice Form** (`CustomerChoice` class):

    - **Constructor**: Initializes the customer choice form.
    - **CustomerChoice_Load Event Handler**:
        - **Database Connection**: Fetches menu details including updated prices using a SQL join query.
        - **Display Menu**: Binds the fetched data to `dataMenu`.
    - **btnReservationChoice_Click Event Handler**: Opens the `Table` form for table reservations.
    - **btnDeliveryChoice_Click Event Handler**: Opens the `Delivery` form for delivery reservations.

4. **LoginForm** (`LoginForm` class):
    - **Constructor**: Initializes the login form.
    - **btnLogin_Click Event Handler**:
        - **Database Connection**: Checks the user's credentials against the `staff` or `customer` tables.
        - **Staff Login**: If `chBoxStaff` is checked, verifies staff credentials.
        - **Customer Login**: If `chBoxStaff` is not checked, verifies customer credentials.
        - **Successful Login**: Shows the `CustomerChoice` form if credentials are correct.
    - **labelGoToSignUp_Click Event Handler**: Opens the `SignUp` form.
    - **chBoxShowPassword_CheckedChanged Event Handler**: Toggles the visibility of the password.

Key Points

- **Database Interactions**: The application frequently interacts with a SQL Server database to fetch and insert data.
- **User Interface**: Utilizes Windows Forms to create the GUI for different user roles and functionalities.
- **Event Handling**: Various event handlers manage user interactions like button clicks and form loading.

SignUp Form

1. **SignUp Form** (`SignUp` class):
    - **Constructor**: Initializes the sign-up form.
    - **Event Handlers**:
        - **btnStaffSignUp_Click**: Opens the `StaffSignUp` form when the staff sign-up button is clicked.
        - **btnCustomerSignUp_Click**: Opens the `CustomerSignUp` form when the customer sign-up button is clicked.
        - **labelGoToLogin_Click**: Opens the `LoginForm` when the label is clicked, allowing users to navigate to the login form.

- **SignUp_Load**: Currently an empty event handler for when the form loads.

StaffSignUp Form

2. **StaffSignUp Form** (`StaffSignUp` class):
    - **Constructor**: Initializes the staff sign-up form.
    - **Event Handlers**:
        - **labelGoToLogin_Click**: Opens the `LoginForm` when the label is clicked, allowing users to navigate back to the login form.
        - **btnSignUp_Click**: Handles the staff sign-up process.
            - **Database Connection**: Connects to the SQL Server database.
            - **Generate New Staff ID**: Queries the database for the latest `staff_id` and increments it to generate a new ID.
            - **Password Confirmation**: Ensures the password and confirm password fields match.
            - **Insert New Staff**: Inserts the new staff details into the `staff` table.
            - **Success Notification**: Displays messages indicating the number of affected rows and the new staff ID.
            - **Navigation**: Opens the `LoginForm` upon successful sign-up.
        - **chBoxShowPassword_CheckedChanged**: Toggles the visibility of the password.
        - **chBoxShowConfirmPassword_CheckedChanged**: Toggles the visibility of the confirm password.

Key Points

- **Navigation Between Forms**: The forms have buttons and labels that allow users to navigate between sign-up, login, and other related forms.
- **Database Operations**: The forms use SQL commands to interact with the database, specifically for querying and inserting data.
- **User Input Validation**: The sign-up forms include basic validation, such as checking that password and confirm password fields match.
- **UI Elements**: The forms contain various UI elements like text boxes, buttons, and labels for user interaction.

Success Form

1. **Success Form** (`Success` class):
    - **Constructor**: Initializes the success form.
    - **Event Handlers**:
        - **btnGoToHome_Click**: When the "Go to Home" button is clicked, it creates an instance of the `CustomerChoice` form, hides the current form, and shows the `CustomerChoice` form.
    - **Success_Load**: Currently an empty event handler for when the form loads.

Key Points

- **Navigation to Home**: The form provides a button (`btnGoToHome`) that allows users to navigate back to the `CustomerChoice` form after a successful operation.
- **UI Elements**: Contains UI elements such as buttons to facilitate user interaction and navigation.
- **Form Initialization**: Uses `InitializeComponent()` to set up the form's components.

Code Flow

1. **Form Initialization**:
    - The `Success` form is initialized and its components are set up when an instance of the form is created.
2. **Button Click Event**:
    - When the "Go to Home" button is clicked, the `btnGoToHome_Click` event handler is triggered.
    - A new instance of `CustomerChoice` is created.
    - The current `Success` form is hidden using `this.Hide()`.
    - The `CustomerChoice` form is shown using `customerChoice.Show()`.

Table Form

1. **Table Form** (`Table` class):

- **Fields**:
    - `food`: Stores the name of the food item.
    - `number`: Stores the customer's phone number.
- **Constructor**: Initializes the form with food and number parameters, and sets up the form components.
- **Event Handlers**:
    - **btnReserve_Click**: Handles the process of reserving a table when the "Reserve" button is clicked.

Key Points

- **Form Initialization**: The form is initialized with the customer's phone number and the selected food item.
- **Database Operations**: The form performs several database operations to process the reservation.
    - **Validation**: Validates the reservation date.
    - **Food Retrieval**: Retrieves the food ID and price from the database.
    - **Order ID Retrieval**: Gets the next available order ID from the database.
    - **Order Insertion**: Inserts a new order into the `order_customer` table.
    - **Table Reservation**: Inserts a new reservation into the `table_reservation` table.

Code Flow

1. **Form Initialization**:
    - The `Table` form is initialized with the customer's number and the selected food item, and the form components are set up.
2. **Reservation Process**:
    - **Validate Reservation Date**: Checks if the reservation date entered by the user is valid. If not, it shows an error message.
    - **Retrieve Food Details**: Connects to the database to retrieve the `food_id` and `price` for the selected food item.
        - If the food item is not found, it shows an error message.
    - **Retrieve Order ID**: Connects to the database to get the next available order ID.

- **Insert Order**: Inserts a new order into the `order_customer` table with the retrieved food details, reservation date, and customer number.
- **Insert Table Reservation**: Inserts the reservation details into the `table_reservation` table, including the number of people and reservation date.

3. **Success Notification**:

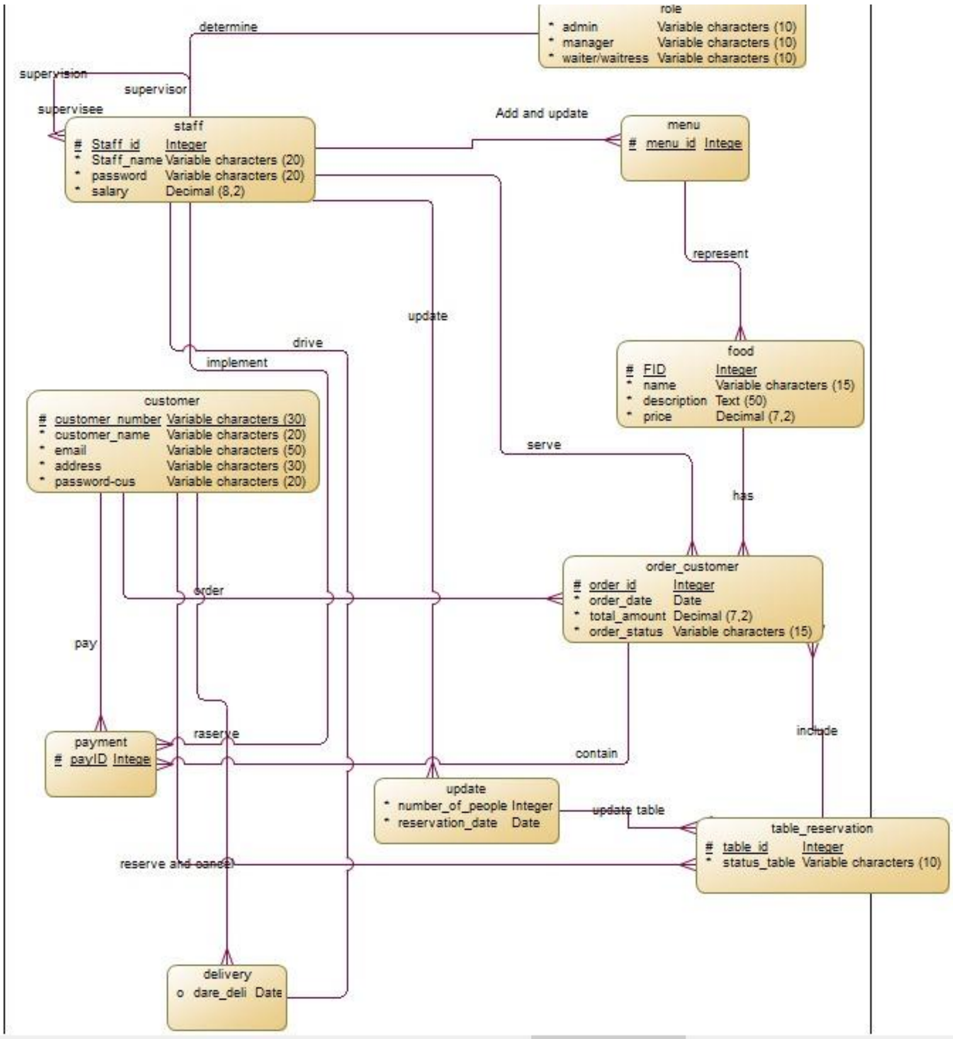
- Shows a success message upon successful table reservation.
- Creates an instance of the `Success` form, hides the current form, and shows the `Success` form.

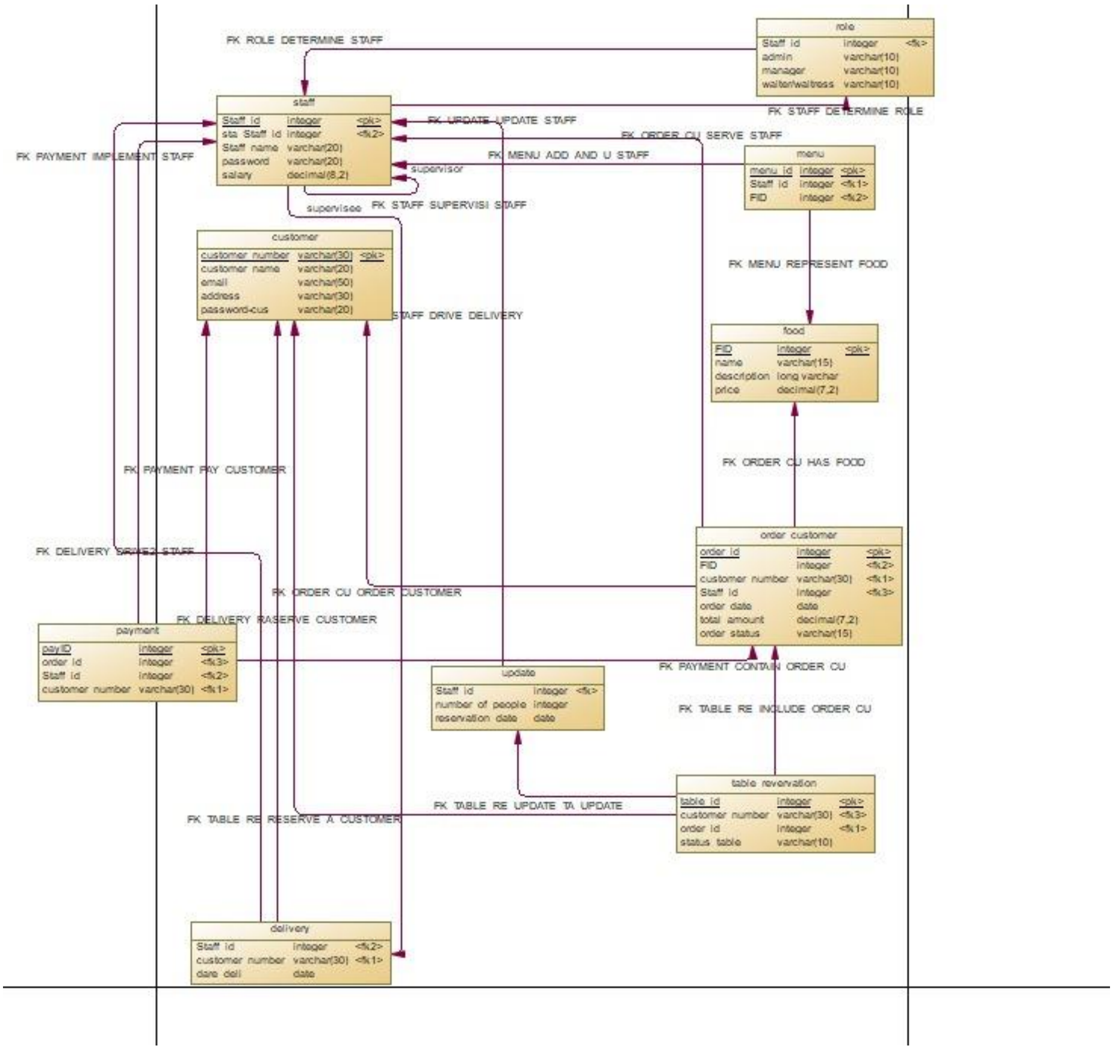
Example Usage

1. **User Input**: The user enters their phone number, the food item they want to order, and the reservation date.
2. **Click Reserve**: The user clicks the "Reserve" button.
3. **Database Operations**:

- The form validates the reservation date.
- It retrieves the food details and the next order ID from the database.
- It inserts the new order and table reservation into the database.

4. **Success Message**: The form displays a success message and navigates to the `Success` form.


role
* admin Variable characters (10)
* manager Variable characters (10)
* waiter/waitress Variable characters (10)
determine
supervision
supervisor
supervisee
staff
# Staff_id Integer
* Staff_name Variable characters (20)
* password Variable characters (20)
* salary Decimal (8,2)
Add and update
menu
# menu_id Intege
represent
update
drive
implement
food
# FID Integer
* name Variable characters (15)
* description Text (50)
* price Decimal (7,2)
customer
# customer_number Variable characters (30)
* customer_name Variable characters (20)
* email Variable characters (50)
* address Variable characters (30)
* password-cus Variable characters (20)
serve
has
order_customer
# order_id Integer
* order_date Date
* total_amount Decimal (7,2)
* order_status Variable characters (15)
order
pay
include
payment
# payID Intege
raserve
contain
update
* number_of_people Integer
* reservation_date Date
update table
table_reservation
# table_id Integer
* status_table Variable characters (10)
reserve and cancel
delivery
o dare_deli Date


FK ROLE DETERMINE STAFF
role
Staff id integer <fk>
admin varchar(10)
manager varchar(10)
waiter/waitress varchar(10)
FK STAFF DETERMINE ROLE
staff
Staff id integer <pk>
sta Staff id integer <fk2>
Staff name varchar(20)
password varchar(20)
salary decimal(8,2)
FK UPDATE UPDATE STAFF
FK ORDER CU SERVE STAFF
FK MENU ADD AND U STAFF
supervisor
supervisee
FK STAFF SUPERVISI STAFF
FK PAYMENT IMPLEMENT STAFF
menu
menu id integer <pk>
Staff id integer <fk1>
FID integer <fk2>
FK MENU REPRESENT FOOD
customer
customer number varchar(30) <pk>
customer name varchar(20)
email varchar(50)
address varchar(30)
password-cus varchar(20)
STAFF DRIVE DELIVERY
food
FID integer <pk>
name varchar(15)
description long varchar
price decimal(7,2)
FK ORDER CU HAS FOOD
FK PAYMENT PAY CUSTOMER
FK DELIVERY DRIVE2 STAFF
order customer
order id integer <pk>
FID integer <fk2>
customer number varchar(30) <fk1>
Staff id integer <fk3>
order date date
total amount decimal(7,2)
order status varchar(15)
FK ORDER CU ORDER CUSTOMER
FK DELIVERY RASERVE CUSTOMER
payment
payID integer <pk>
order id integer <fk3>
Staff id integer <fk2>
customer number varchar(30) <fk1>
FK PAYMENT CONTAIN ORDER CU
update
Staff id integer <fk>
number of people integer
reservation date date
FK TABLE RE INCLUDE ORDER CU
table revervation
table id integer <pk>
customer number varchar(30) <fk3>
order id integer <fk1>
status table varchar(10)
FK TABLE RE UPDATE TA UPDATE
FK TABLE RE RESERVE A CUSTOMER
delivery
Staff id integer <fk2>
customer number varchar(30) <fk1>
dare deli date