**Sweet Delights Cake Shop Project**

In this activity, you will test your programming and database management skills by creating a functional application for a local cake shop. This project will give you hands-on experience building a complete system from the ground up, combining front-end user interface design with back-end database logic.

Throughout this project, you will take on the role of a software developer. You will be responsible for building a program to manage customer details, process new orders, and track products like cakes, icings, and toppings.

This activity is broken down into four key parts:

1. Database Design and Creation
2. User Interface (UI) Design
3. Backend Logic and Data Connection

By the end of this project, you will have a fully working application and a deeper understanding of how modern software systems are built.

**Part 1: Database Design and Creation**

In this part of the activity, you will create the database and all the necessary tables for the Sweet Delights Cake Shop application using Microsoft SQL Server. This is the backbone of our program, where all the customer, order, and product information will be stored.

**Relational Schema**

Here is the complete relational schema you will use to design your database. Pay close attention to the primary keys and how they will create relationships between tables.

* *Customer(CustomerID, FirstName, LastName, ContactNumber, Email, StreetAddress, Barangay, City)*
* *Order(OrderID, CustomerID, OrderDate, OrderType, TotalAmount, Status)*
* *Cake(CakeID, CakeType, Price)*
* *Icing(IcingID, IcingType, price)*
* *Topping(ToppingID, ToppingType, price)*
* *Customization(CustomizationID, OrderID, CakeID, PersonalizedMessage, MixedFlavors, SpecialInstructions)*
* *Billing(BillingID, OrderID, PaymentMethod, AmountPaid, PaymentDate)*
* *Delivery(DeliveryID, OrderID, DeliveryType, DeliveryDate, DeliveryTime, StreetAddress, Barangay, City)*
* *OrderDetails(OrderID, CakeID, Quantity, Subtotal, PRIMARY KEY(OrderID, CakeID))*
* *CakeIcing(CakeID, IcingID, PRIMARY KEY(CakeID, IcingID))*
* *CakeTopping(CakeID, ToppingID, PRIMARY KEY(CakeID, ToppingID))*

**Tasks**

Use **Microsoft SQL Server Management Studio (SSMS)** to complete the following tasks.

**Task 1: Create the Database**

* Create a new database and name it SweetDelightsDB.

**Task 2: Create the Tables**

* Create each of the eleven tables from the relational schema, ensuring you use the correct data types for each field.
* Mark the primary keys and set up any auto-increment properties as needed.

**Task 3: Define Relationships (Foreign Keys)**

* Establish the relationships between the tables by defining foreign keys. For example:
  + CustomerID in the Order table should reference the CustomerID in the Customer table.
  + OrderID in the OrderDetails table should reference the OrderID in the Order table.
  + CakeID in the OrderDetails table should reference the CakeID in the Cake table.
  + CakeID in CakeIcing should reference CakeID in Cake, and IcingID in CakeIcing should reference IcingID in Icing.
  + Do the same for the CakeTopping, Customization, Billing, and Delivery tables.

**Task 4: Populate with Sample Data**

* Insert at least five sample records into the Cake, Icing, and Topping tables using your data. This will be the data you use to test your program in later parts.

**Part 2: User Interface (UI) Design**

In this part, you will use Visual Basic to design the front-end user interface for the Sweet Delights Cake Shop application. For now, you will only create the visual layout and controls; no code is needed in this part.

**Tasks**

Use **Visual Studio** and **Visual Basic** to complete the following tasks.

**Task 1: Create the Main Application Form**

* Create a new Windows Forms Application project in Visual Studio.
* Design a main form that will serve as the application's dashboard. This form should be the starting point for the user.
* Add buttons or menu items to the form that will allow a user to navigate to other parts of the application, specifically to:
  + "Manage Customers"
  + "Manage Orders"
  + "View Cakes, Icings, and Toppings"

**Task 2: Design the Customer Management Form**

* Create a separate form dedicated to managing customer information. Name this form something like frmCustomer.
* On this form, include the following controls for **CRUD** (Create, Read, Update, Delete) operations:
  + Text Boxes: One for each field in the Customer table (e.g., txtFirstName, txtLastName, txtContactNumber, etc.).
  + Buttons: Add buttons for Save (or Add), Update, Delete, and Clear.
  + Data Grid View: Include a DataGridView control to display all existing customer records from the database. This will be where users can view and select customers to edit or delete.

**Task 3: Design the Order Management Form**

* Create a separate form named frmOrder to manage customer orders. This form will be more complex and should contain the following controls:
  + Customer Selection: A control (like a ComboBox) to select an existing customer for the order.
  + Order Details Grid: A DataGridView is used to show the details of the current order, including cake type, icing type, topping type, quantity, and subtotal.
  + Product Selection: ComboBox controls for selecting a Cake, Icing, and Topping.
  + Quantity Selector: A control (e.g., a numeric up-down or text box) for entering the quantity of an item.
  + Order Buttons: These are the buttons for adding, removing, and placing orders.
  + Total Display: A label or text box to display the total amount of the order.

***Note:*** *Remember to give your forms and controls clear and descriptive names (e.g., btnSave, dgvCustomers, txtOrderDate). This practice is crucial for making your code easy to read and understand in the next part.*

**Part 3: Backend Logic and Data Connection**

In this part, you will write the Visual Basic code to connect the user interface you designed in Part 2 to the database you created in Part 1. You will implement the functionality to save, retrieve, update, and delete data.

**Tasks**

**Task 1: Establish Database Connection**

* In your Visual Basic project, add a module or a class to handle the database connection.
* Write a function to create and return a new database connection object. This function should contain a connection string that points to your SweetDelightsDB database on your local SQL Server instance.
* Make sure to handle potential exceptions (e.g., if the connection fails) using a Try...Catch block.

**Task 2: Implement Data Retrieval and Display**

* Go to the frmCustomer form.
* In the form's Load event, write a Sub routine to retrieve all customer records from the Customer table.
* Use a data adapter to fill a DataTable or DataSet with the customer data.
* Bind the DataTable to your dgvCustomers DataGridView control to display the information to the user.
* Do the same for any other forms where you must display a list of items from the database.

**Task 3: Implement Customer CRUD Operations**

* Create/Add: On the btnSave button's Click event, write code to construct an INSERT SQL query to add a new customer record.
* Update: On the btnUpdate button's Click event, write a routine to construct an UPDATE SQL query using the CustomerID in the WHERE clause.
* Delete: On the btnDelete button's Click event, write a routine to construct a DELETE SQL query using the CustomerID in the WHERE clause.
* Clear: On the btnClear button's Click event, write a routine to clear all text boxes and reset the form.
* After each operation (Save, Update, Delete), call the data retrieval function to refresh the DataGridView and show the updated data.

**Task 4: Implement Order Management Logic**

* Product Selection: On the frmOrder form, populate the ComboBox controls for Cakes, Icings, and Toppings by retrieving data from their respective database tables.
* Add Item: On the btnAddItem button's Click event, write code to calculate the subtotal for the selected cake and its customizations. Add this new item as a row in the dgvOrderDetails DataGridView.
* Calculate Total: As items are added to the order, automatically update the TotalAmount label by summing the subtotal of each item in the DataGridView.
* Place Order: On the btnPlaceOrder button's Click event, create a routine that handles the most complex part of the application:
  + Insert a new record into the Order table.
  + For each item in the DataGridView, insert a record into the OrderDetails table.
  + Insert a new record into the Billing table.
  + Insert a new record into the Delivery table.