Test Project Session 5

IT Software Solution for Business

Independent Test Project Designer: Ramin Mohammaddoust

Independent Test Project Validator: Afshin Dehghani

Introduction

In this session, you will enhance the existing Belle Croissant Lyonnais desktop application by adding more features. You'll work with the product/order API and a new database schema to implement promotions, manage loyalty programs, and handle ingredient inventory and recipes.

This session will assess your skills in:

- **Advanced Development:** Building complex features and integrating them with existing systems.
- **API Integration:** Interacting with the provided API to fetch and update data.
- **Data Manipulation:** Working with data from the database to implement business logic.
- **Problem-Solving:** Finding and resolving issues that arise during development.

You will be provided with all the necessary resources, including the API documentation and the database schema. Your goal is to demonstrate your ability to create a functional and user-friendly application that meets the needs of Belle Croissant Lyonnais.

Contents

This session package includes the following materials:

1. **Session Instructions (PDF):** Detailed instructions outlining the tasks to be completed and deliverables expected for this session.
2. **Common Folder:** This folder contains additional resources such as the Belle Croissant Lyonnais logo, icons, style guide, and other design assets that can be used throughout the development of the application.
3. **API Documentation (PDF):** Documentation for the product/customer/order API, providing details about endpoints, request/response formats, and authentication.
4. **Database Schema (SQL):** A SQL script containing the structure for the tables Promotions and LoyaltyProgram, which you will use in this session.

Description of Project and Tasks

In this session, you will enhance the Belle Croissant Lyonnais desktop application with advanced features.

Guidelines:

1. **Easy to Use:** Make the interface simple and easy for staff to understand.
2. **Looks Good:** Follow the Belle Croissant Lyonnais Style Guide for the design.
3. **Works Well:** Check that all parts of the application work correctly and without errors.
4. **Secure:** Protect customer data and ensure the application is safe to use.
5. **On Time:** Finish all tasks within the time limit.

Technical Considerations:

1. **Database Interaction:** Use the provided database schema and SQL queries to interact with the database.
2. **API Integration:** Integrate with the customer/product/order API to fetch necessary data.
3. **Data Validation:** Ensure user input is correct and complete.
4. **Error Handling:** Display clear messages to the user if there are any problems.

Additional Considerations:

- The application should work smoothly and quickly on the provided virtual machine.
- Use clear labels and instructions for all UI elements.
- Organize information in a way that is easy for staff to understand.
- Consider edge cases and potential errors in user input and data handling.

Instructions to the Competitor

5.1 Promotion and Loyalty Program Database Implementation

Objective:

Make database tables to store info about special offers and customer rewards for Belle Croissant Lyonnais.

Tasks:

1. **Create Tables:**
    - Use the SQL code given (Session5_Database_Schema.sql) to create tables in the database.
2. **Add Promotion Data:**
    - Put at least 10 different special offers into the Promotions table.
    - The offers should be different types (discount %, fixed amount off), for different products, and last for different times with at least two offers covering multiple products.
    - Some offers can be at the same time.
    - Make one offer with an end date that comes before the start date.
3. **Add Loyalty Data:**
    - Put data for all customers in the LoyaltyProgram table. Start everyone with 0 points.
    - Give 250 randomly chosen customers random point values (between 50 and 1500) to indicate they are active members.

Deliverables:

- **Database Access:** Give the login info for your database (server name, database name, username, password). You can put this in a text file named Session5_DatabaseCredentials.txt.

- **Check Script:** Write a SQL script (Session5_DatabaseVerification.sql) to check:
    - Tables exist and are correct.
    - How many rows are in each table.
    - First 5 rows of each table to see if the data is right.
    - Find the offer with the wrong date.
    - Find customers with starting points.

5.2 Promotion Management UI

Objective:

Develop a standalone desktop application for Belle Croissant Lyonnais to manage promotions, including customizable quantity-based discounts and conflict resolution.

Tasks:

1. **Design and Implement the UI:**
    - Create a user interface for managing promotions within a new desktop application.
    - The interface should include the following elements:
        - A table to display existing promotions, with columns for:
            - Promotion ID (auto-generated and hidden from the user)
            - Promotion Name
            - Discount Type (percentage or fixed amount)
            - Discount Value
            - Applicable Products (multi-select list populated from the Products table)
            - Start Date
            - End Date
            - Minimum Order Value (optional)
            - Priority (numeric input, higher number indicates higher priority)
        - Input fields for adding or editing promotion details (with appropriate validation).
        - Buttons for:
            - Adding a new promotion
            - Saving changes to an existing promotion
            - Deleting a promotion
2. **Database Interaction:**
    - Implement the logic to:

- Fetch promotion data from the Promotions table in the database and populate the table in the UI.
- Insert new promotions into the Promotions table when the "Add" button is clicked.
- Update existing promotions in the Promotions table when the "Save" button is clicked.
- Delete promotions from the Promotions table when the "Delete" button is clicked (with confirmation).

  - Handle any database errors gracefully and display appropriate error messages to the user.

3. **User-Guided Resolution Wizard:**

   Develop a step-by-step wizard to guide the user through conflict resolution when adding or editing promotions, specifically addressing scenarios where a product has the same partial or complete date period in common with another promotion of the same priority.

   - **Step 1: List of Conflicts**
     - **Scenario:** Multiple conflicts may exist, involving different products and date ranges.
     - **Action:** The wizard presents a clear list of all identified conflicts, summarizing each conflict with the following details retrieved from the database:
       - **Conflicting Promotions:** PromotionName of the promotions involved.
       - **Products:** The specific ApplicableProducts (product IDs) causing the conflict.
       - **Date Ranges:** The overlapping StartDate and EndDate for each promotion.

   - **Step 2: Conflict Notification**
     - **Scenario:** The user selects a specific conflict from the list to address.
     - **Action:** The wizard displays a detailed explanation of the selected conflict, emphasizing that the conflict exists because a product is associated with multiple promotions of the same priority within the same partial or complete date period:
       - **Product and Date Conflict:** The wizard explicitly highlights the specific ApplicableProducts (product IDs) and the overlapping StartDate and EndDate that constitute the conflict.
       - **Priority Emphasis:** The wizard also indicates that the conflicting promotions share the same Priority level.
       - **Visual Aid:** A timeline is provided to illustrate the overlap of the product's presence across promotions based on the StartDate and EndDate.

   - **Step 3: Priority Change**
     - **Scenario:** The user chooses to explore adjusting promotion priorities to potentially resolve the conflict.
     - **Action:**
       - The wizard explains how Priority affects promotion application and stacking, emphasizing that priority changes are unlikely to resolve conflicts where products are active in multiple promotions with the same priority simultaneously.
       - A list of available priority levels is presented, with the current Priority highlighted.
       - The user can select a higher Priority level from the list.

        - If a new priority is selected, the corresponding PromotionId's Priority value is updated in the database.
        - The wizard warns if changing priority might impact other promotions or have unintended consequences.

- **Step 4: Date/Product Adjustment**
    - **Scenario:** The conflict can be resolved by either adjusting the dates of one or more promotions to eliminate overlap OR by removing the conflicting product(s) from one of the promotions.
    - **Action:**
        - **Date Adjustment:** The wizard suggests alternative StartDate and/or EndDate values that would avoid the overlap for the conflicting ApplicableProducts. The user can either select one of the suggested ranges or manually adjust the dates. The corresponding StartDate and/or EndDate values in the Promotions table are updated accordingly.
        - **Product Removal:** If date adjustment is not feasible or desirable, the wizard presents a list of the conflicting ApplicableProducts within the selected promotion. The user can select one or more products to remove from the promotion's ApplicableProducts list in the database.
        - **Real-time Feedback:** As the user makes adjustments, the wizard provides real-time feedback on whether the changes resolve the conflict by checking against the updated data in the Promotions table.

- **Step 5: Cancellation**
    - **Scenario:** The user decides to abandon the new/modified promotion.
    - **Action:** A confirmation prompt asks the user to confirm the cancellation. If confirmed, no changes are made to the Promotions table in the database.

- **Iteration:**

  The user can repeat steps 2-6 for each conflict in the list until all conflicts are resolved or the new/modified promotion is canceled.

- **Completion:**

  Once all conflicts are resolved, the user can proceed with saving the new/modified promotion.

Deliverables:

- **File Name:** Session5_PromotionsApp.exe
- **File Type:** Executable file of the standalone desktop application for promotion management.

Additional Notes:

- Refer to the "Promotion Management" and "Conflict Resolution Wizard" folders for interactive examples of the expected UI and functionality. These examples use sample data and are meant to illustrate the desired workflow, not to dictate specific design choices.

5.3 Customer Loyalty Management UI

Objective:

Develop a user interface for Belle Croissant Lyonnais' desktop application to manage customer loyalty points, integrating with the main product/order API to fetch customer data and calculate rewards.

Tasks:

1. **Customer Listing and Search:**
    - Display a table of customers fetched from the main API.
    - Table columns: Customer ID, First Name, Last Name, Email, Membership Status, Loyalty Points (from the local database), total_spending.
    - The table should be paginated, showing 10 customers per page initially, sorted by Customer ID in descending order.
    - Implement a search bar that filters the customer list in real-time based on name, ID, or email.
    - Add sorting options for each column (ascending/descending).
2. **Loyalty Points Display and Adjustment:**
    - When a customer is selected from the table, display their details in a separate section or form:
        - Customer ID
        - First Name
        - Last Name
        - Email
        - Membership Status
        - Loyalty Points (editable numeric input field, allow only integers)
    - Add "Save Changes" and "Cancel" buttons to the details section.
        - "Save Changes": Updates the loyalty points in the database and refreshes the customer listing.
        - "Cancel": Discards any changes made to the loyalty points.
3. **Points Calculation:**
    - Add a "Recalculate Points" button to the customer details section.
    - When clicked:
        - Fetch the customer's order history from the main API (/api/orders).
        - Calculate points earned based on the following tiered system:
            - **Basic:** 10 point per 10 euros spent (round down).
            - **Silver:** 12 points per 10 euros spent (round down).
            - **Gold:** 15 points per 10 euros spent (round down).

        - Add 25 bonus points if the current date is the anniversary of the customer's join_date.
        - Add 5 bonus points for each purchase made during a valid promotion period (all tiers).
    - Display a detailed breakdown of the points calculation in a table or dialog, showing points earned per order and any bonuses.
    - "Confirm" button to update the loyalty points in the database and refresh the customer details.
4. **Rewards/Tier Upgrade:**
    - If the customer has 1,000 or more points, display a "Redeem Rewards" button in the customer details section.
    - When clicked, show a dialog with three options:
        - 5 euros discount on their next purchase.
        - 10% discount on their next purchase.
        - Upgrade to the next membership tier (if applicable).
    - If a reward is selected:
        - Deduct 1,000 points from the customer's balance.
        - If a discount is chosen, create a new promotion in the Promotions table with the appropriate discount type and value, applicable to all products, starting on the current date and ending 30 days later, with no minimum order value and high priority (e.g., 99).
        - If a tier upgrade is chosen, update the customer's membership_status in the database.
        - Refresh the customer details to reflect the changes.

Deliverables:

- **File Name:** Session5_LoyaltyApp.exe
- **File Type:** Executable file of the standalone desktop application for loyalty program management.

Additional Notes:

- Refer to the "Loyalty Management" folder for interactive example of the expected UI and functionality. This example uses sample data and is meant to illustrate the desired workflow, not to dictate specific design choices.