CIS 560: Project Report

By Kaitlyn Pritchard, Kylie Phommasack & Tinashe Sekabanja

Introduction

Overview:

The Blossom Jewelry Boutique is an application -based e-commerce platform specializing in handmade jewelry. The system is designed to serve two primary user groups: customers and administrative staff. For customers, the application provides an intuitive interface to browse, search, and purchase unique jewelry items. For administrators, it offers essential inventory management features, such as the ability to add new products and discontinue existing ones. This dual-function system aims to streamline the shopping experience for users while enabling efficient backend operations for store management.

Technical Description
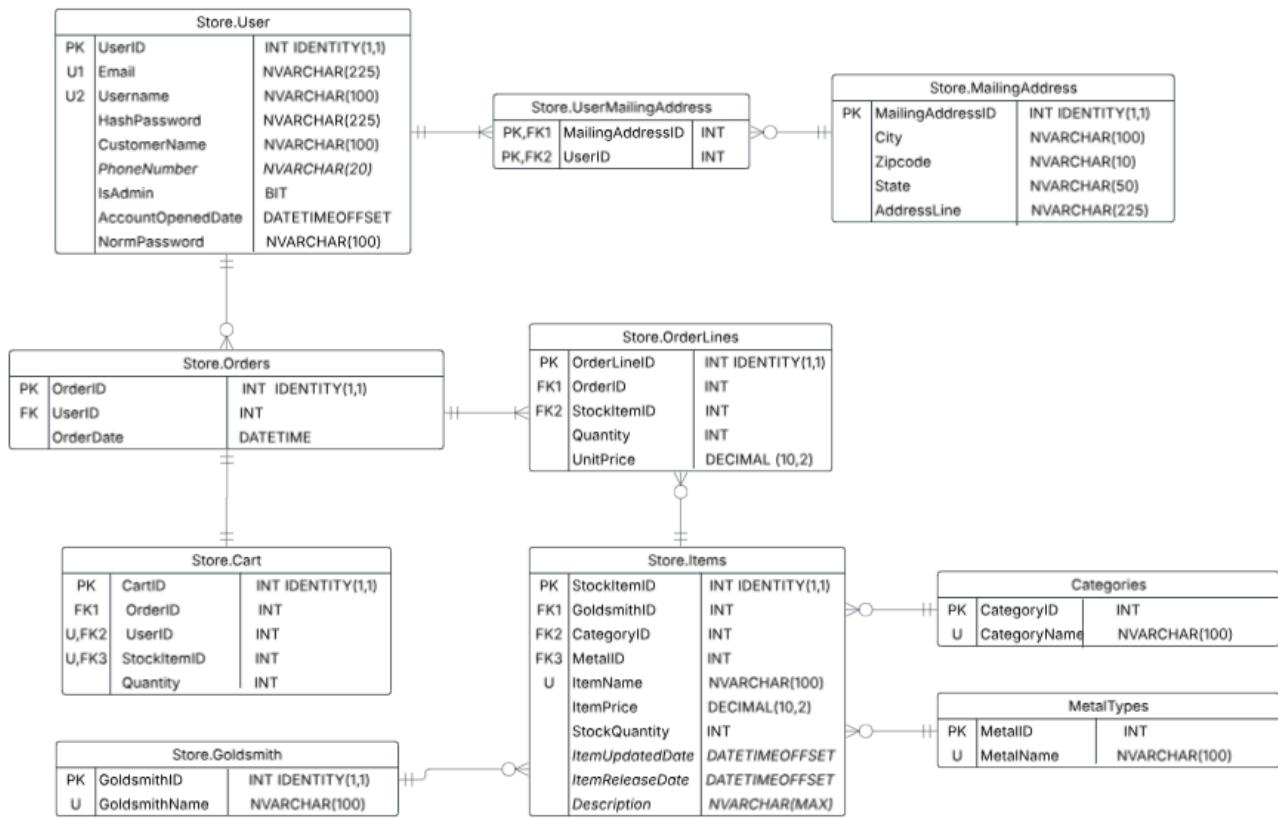
Languages:

C#, SQL - The application was developed using the IDE, Visual Studio 2022, in the language C#. Our Jewelry Boutique program consists of individual forms for each of the user interfaces. All database information is written in SQL.

Libraries:

BCrypt.Net-Next - Whether a user is creating an account or logging in, their password will be stored using this library to hash the password into a string that cannot easily be reversed. This ensures that a user's information is secure in the database.

Physical Database Model


System Design

Clarification:
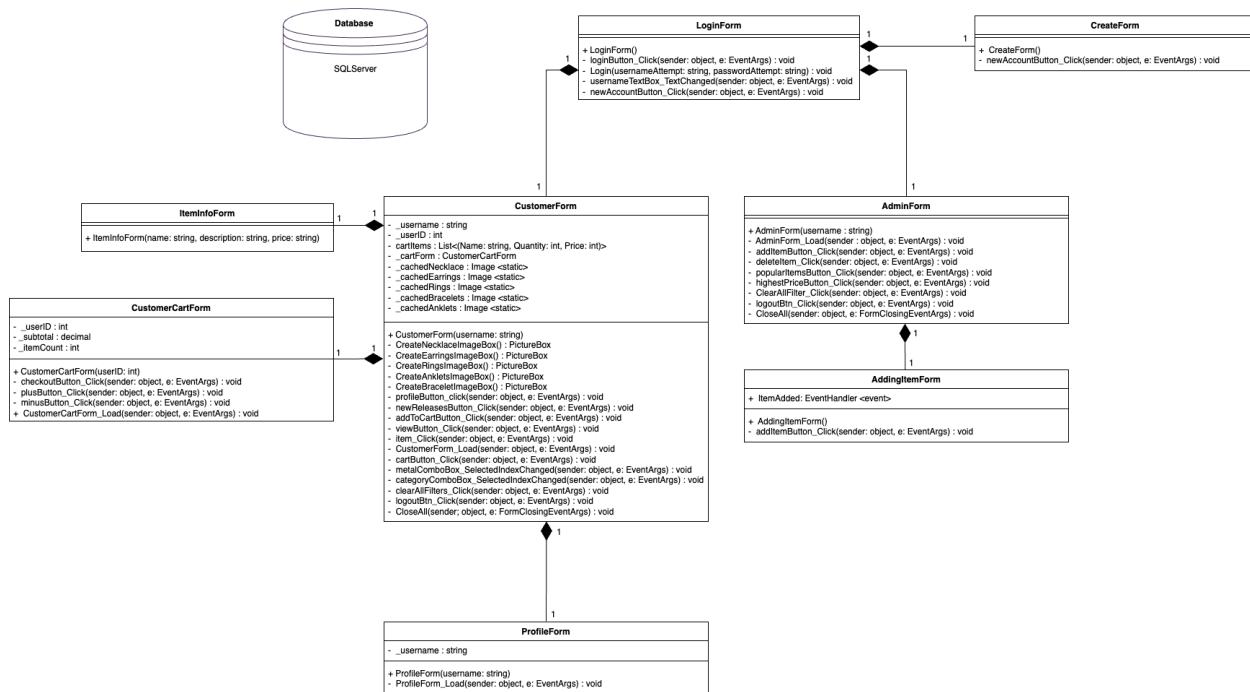
Our project is composed of 8 separate forms and uses the SQL Server to store the databases. When the project starts up, the first form that is created is the "LoginForm." Depending on how the user interacts with this form, it will either open up a "CreateForm," an "AdminForm," or "CustomerForm." If the user is prompted to "AdminForm," they are able to open the "AddingItemForm." If the user is prompted to "CustomerForm," they are able to open up "CustomerCartForm," "ItemInfoForm," and "ProfileForm." On each of these forms, the database is accessed in order to find users, gather product information, save customer sales, or view highest sales made by the goldsmith users.

- *LoginForm*: This form is the starting point of the application. Users will be asked to either sign-in with their username or password or create a new account.
- *CreateForm*: This form allows the user to create a new account. They must also add enter in their mailing address when creating an account.
- *CustomerForm*: This form is the homepage for the customer logged in. Here, the customer can view their profile, view the cart, logout, filter by new release, metal
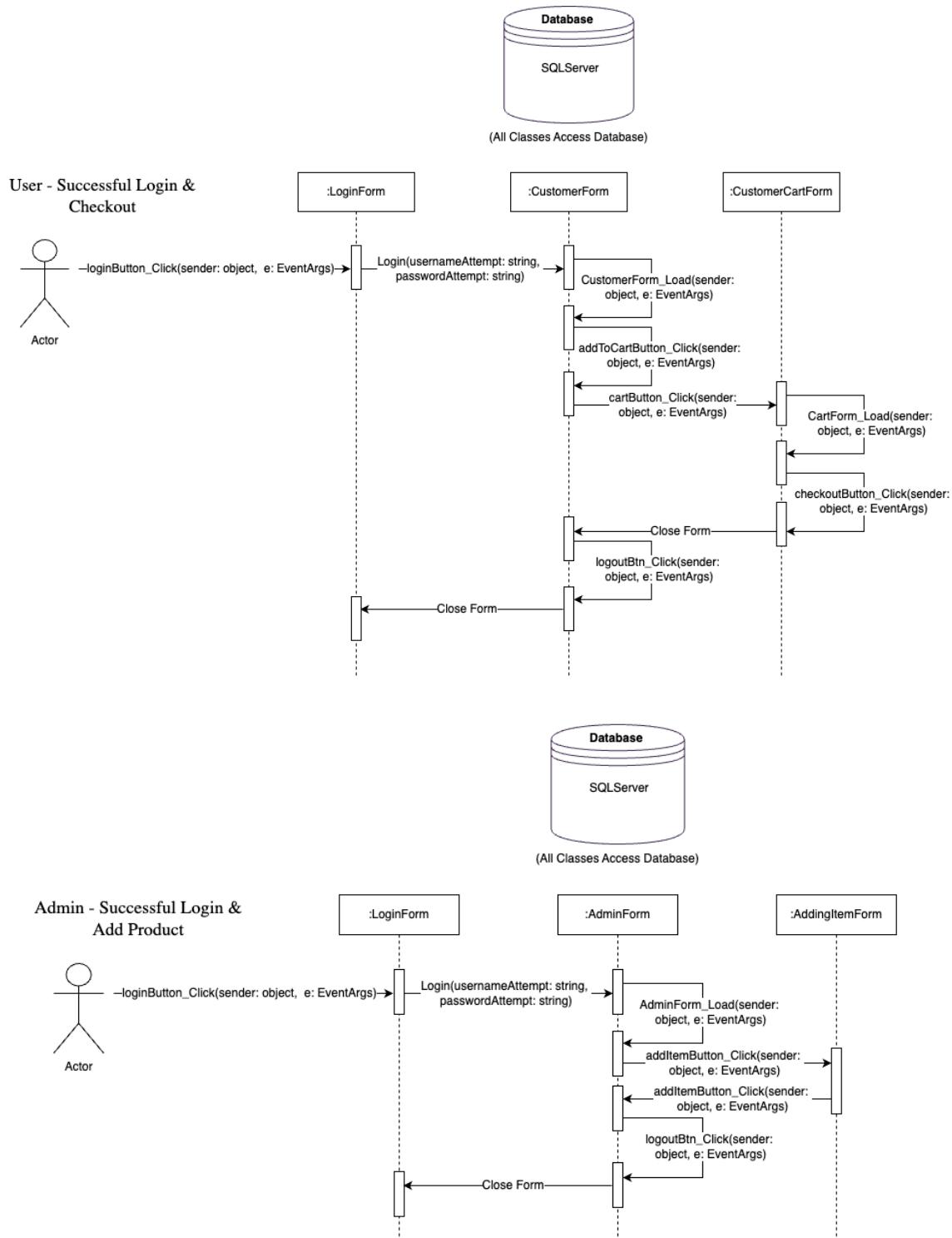
type or category. Also, it displays all the products specified by the filters. The user is able to add the item to their cart and click the item for more details.

- *AdminForm*: This form is the homepage for the admin logged in. Here, the admin can add a product, delete a product, view the most popular goldsmiths, view the customers that have spent the most, filter by the most popular items from the past 3 months, or sort the items by the highest price.
- *CustomerCartForm*: This form is the cart containing all the items that a customer has added to their cart. Users are able to add or subtract from the items. The user can click the 'Checkout' button to complete a successful transaction.
- *ItemInfoForm*: This form consists of the details regarding a product.
- *AddingItemForm*: This form allows the admin to add products that users are able to purchase.
- *ProfileForm*: This form is for the user to be able to view their email, username, and phone number.

UML Class Diagram:

Sequence Diagrams:

Database

SQLServer

(All Classes Access Database)

User - Successful Login & Checkout — :LoginForm — :CustomerForm — :CustomerCartForm

Actor
—loginButton_Click(sender: object, e: EventArgs)→
Login(usernameAttempt: string, passwordAttempt: string)→
CustomerForm_Load(sender: object, e: EventArgs)
addToCartButton_Click(sender: object, e: EventArgs)
cartButton_Click(sender: object, e: EventArgs)→
CartForm_Load(sender: object, e: EventArgs)
checkoutButton_Click(sender: object, e: EventArgs)
←Close Form—
logoutBtn_Click(sender: object, e: EventArgs)
←Close Form—

Database

SQLServer

(All Classes Access Database)

Admin - Successful Login & Add Product — :LoginForm — :AdminForm — :AddingItemForm

Actor
—loginButton_Click(sender: object, e: EventArgs)→
Login(usernameAttempt: string, passwordAttempt: string)→
AdminForm_Load(sender: object, e: EventArgs)
addItemButton_Click(sender: object, e: EventArgs)
addItemButton_Click(sender: object, e: EventArgs)
logoutBtn_Click(sender: object, e: EventArgs)
←Close Form—

System Features & Usage

Customer Experience:
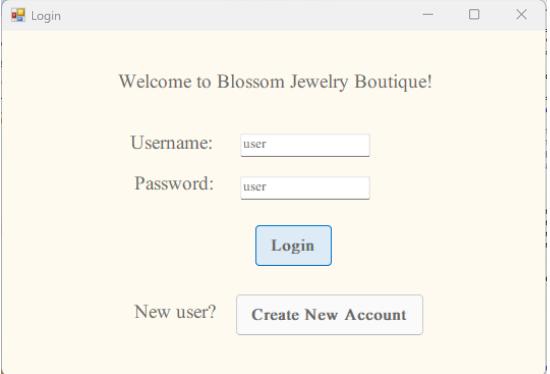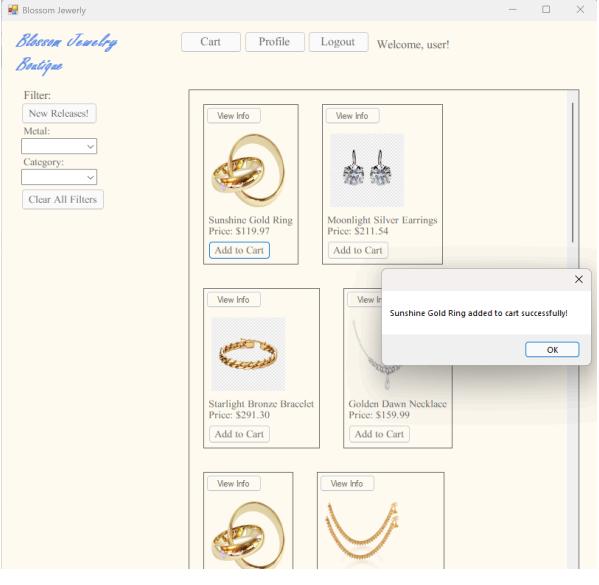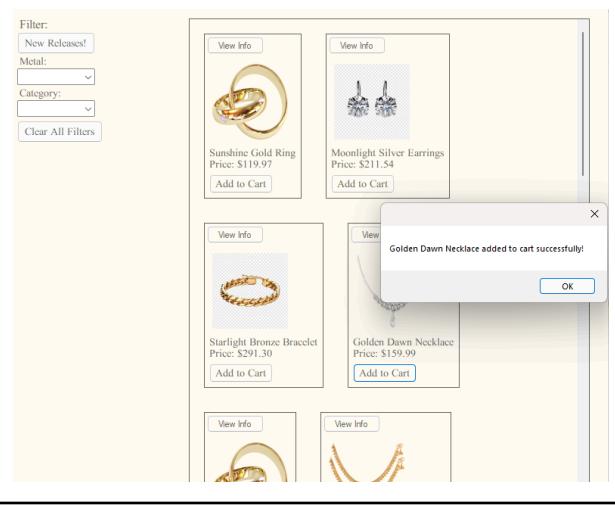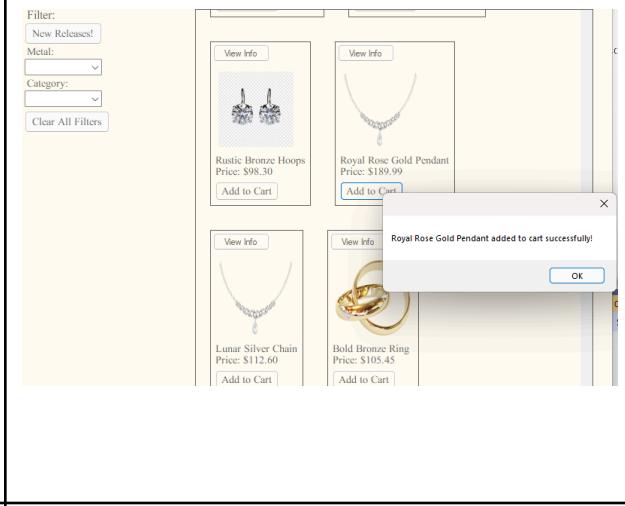
We built a system where as a customer you can view all the products in the inventory. The user can filter the items they want to see by 'category', 'metal type', and 'new releases'. The Admin experience:
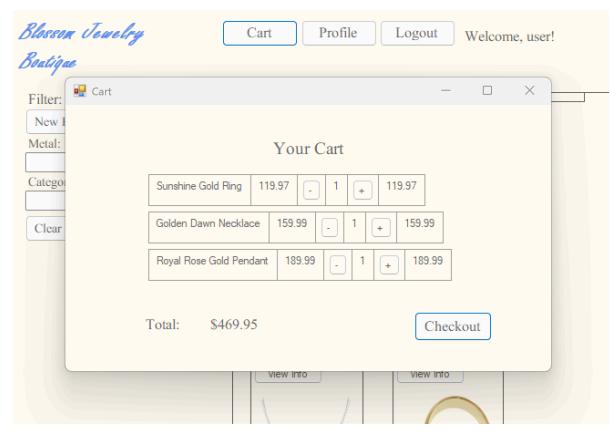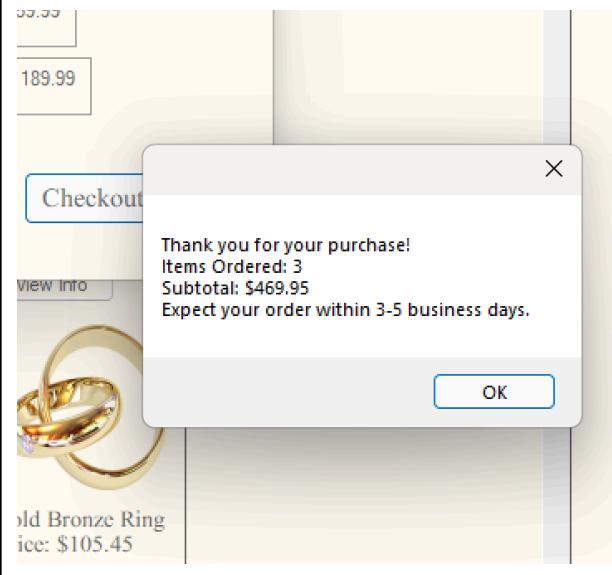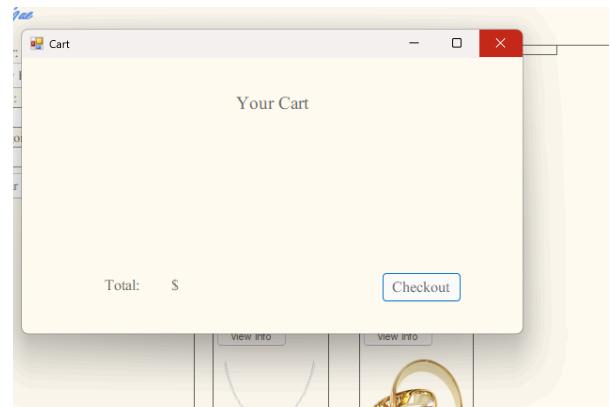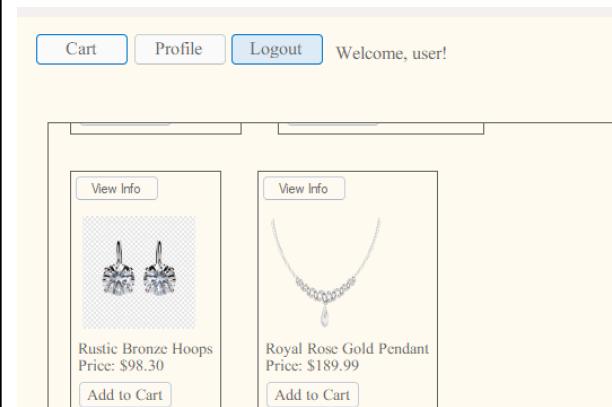
When admins log into the application, they are able to access administrative features. They can generate the most popular items - clicking this button results in a display of the top 10 most popular items with the highest sales.

When the program is initially started, the Login Form page is opened which allows the user to enter their username and password or create a new account. Creating a new account opens a form for the user to enter all of their information and then inserts into the User table. The User information table in our database stores whether each account is an admin or a customer. This decides whether the Customer or Admin Form is opened. When the customer form is opened, all of the products are shown with the option to view more information or add it to the cart. The user can also open a form showing their profile information and a different form showing their cart information. Inside the cart form, the customer can increase and decrease the item quantities and checkout. Checking out deletes the users cart items and inserts an order. If the Admin form is opened instead, the products are again shown but with the option to delete each item. An admin user can also add new items and filter the items by price and most popular. Either the customer or admin can log out and be brought back to the Login Form.

Scenario #1: "A user logs in and successfully checks out after adding 3 different items to their cart."
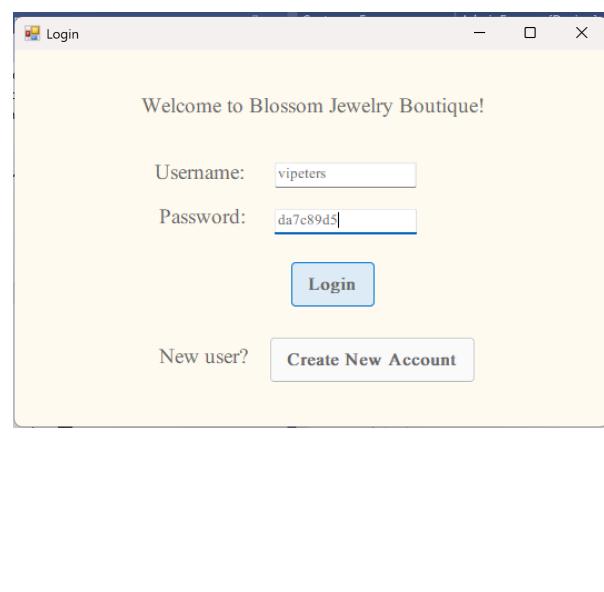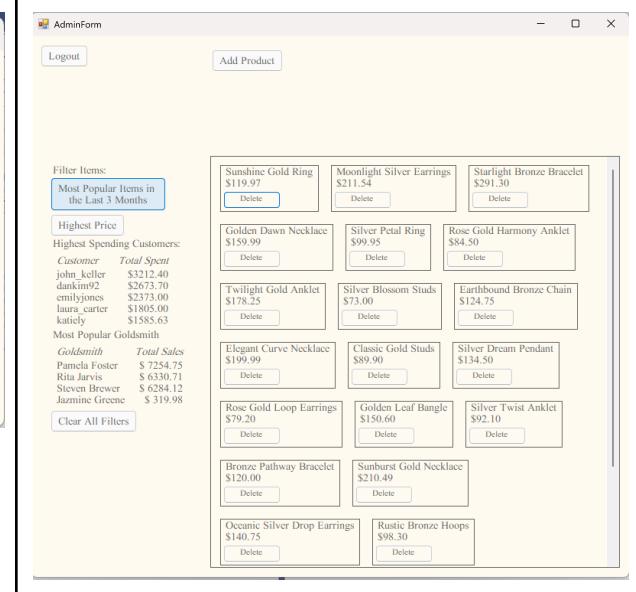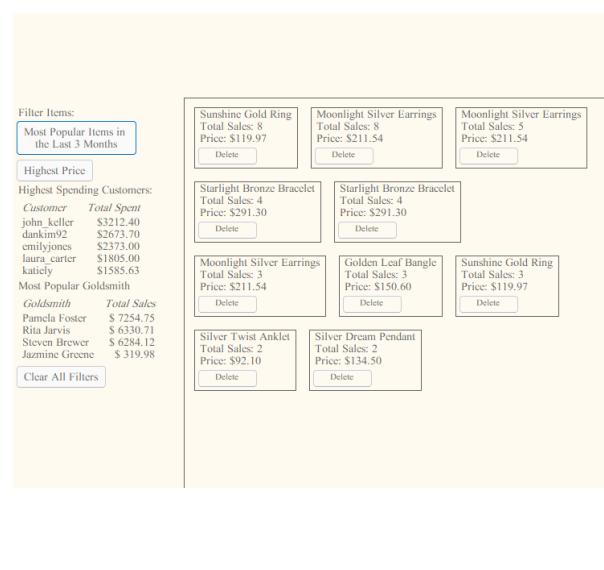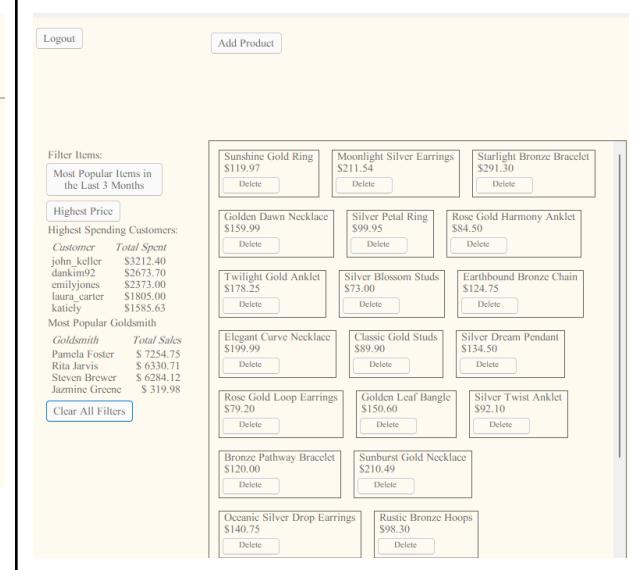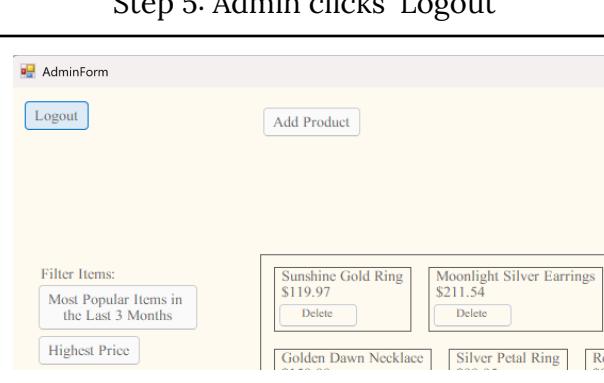
Step 1: User logs in	Step 2: User adds 'Sunshine Gold Ring' to cart

Step 3: User adds 'Golden Dawn Necklace' to cart

Step 4: User adds 'Royal Rose Gold Pendant' to cart

Step 5: User clicks to open up their cart

Step 6: User clicks 'Checkout'
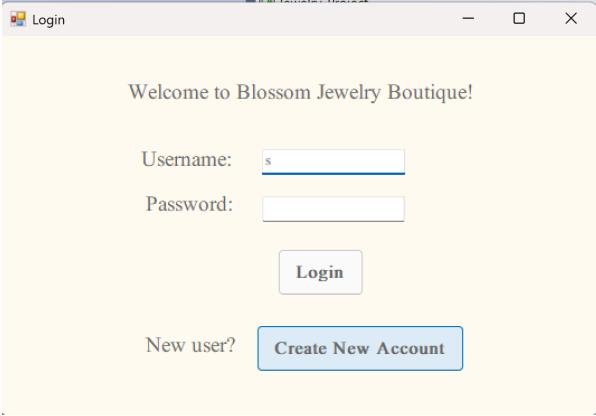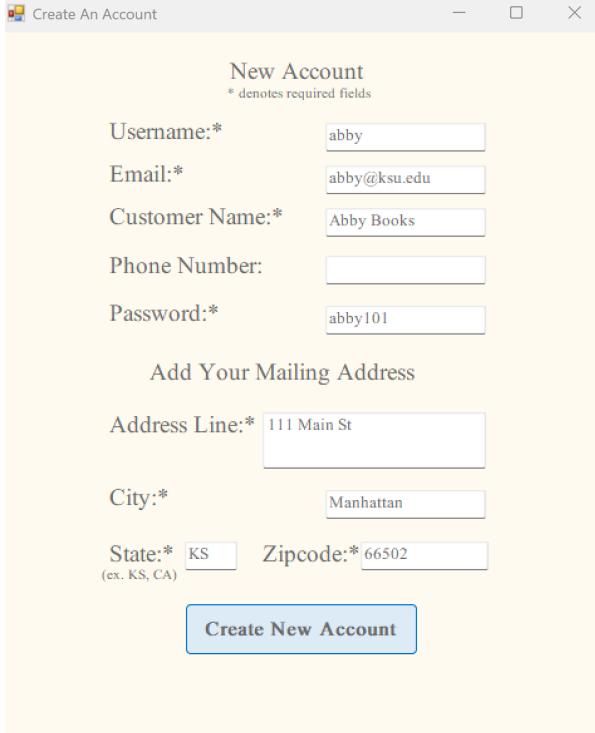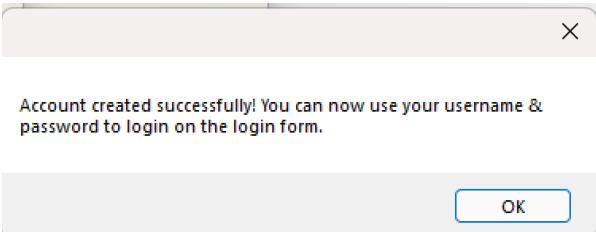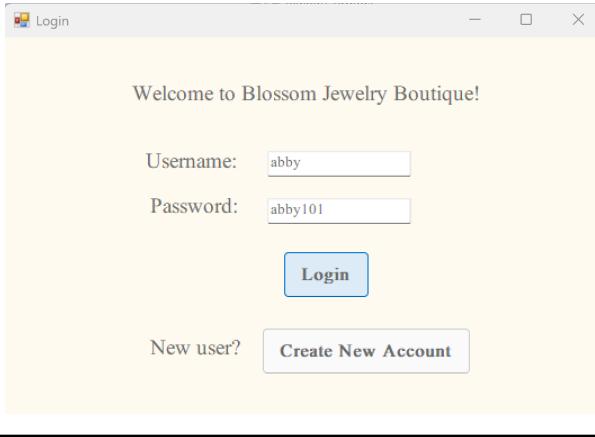
Step 7: User verifies empty cart & closes	**Step 8: User clicks 'Logout'**
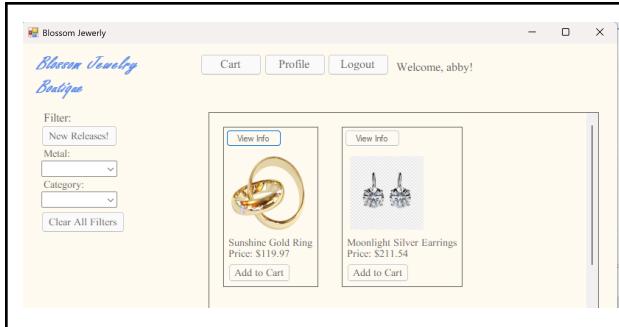
Scenario #2: "An admin logs in and is able to observe the most popular items.

Step 1: Admin logs in	Step 2: Admin clicks to view 'Most Popular Items' on main admin page

Step 3: Admin observes filter results

Step 4: Admin clicks 'Clear All Filters'

Step 5: Admin clicks 'Logout'

Scenario #3: "A user is able to create an account and log in."
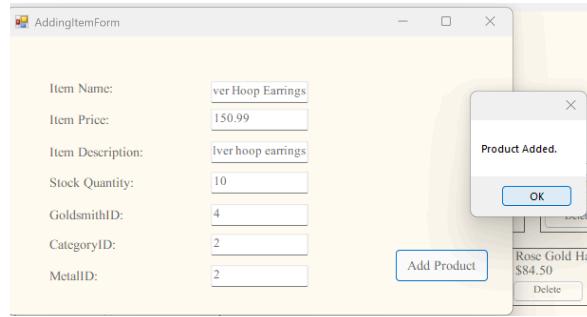
| Step 1: User clicks 'Create New Account'	Step 2: User fills in new account form
Step 3: User gets this message once the form is submitted	**Step 4: User logs in with new account**
Step 5: User verifies their name at top of form	
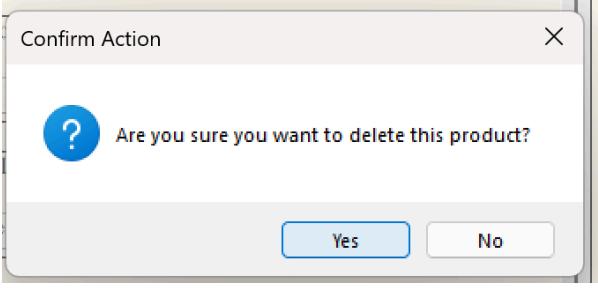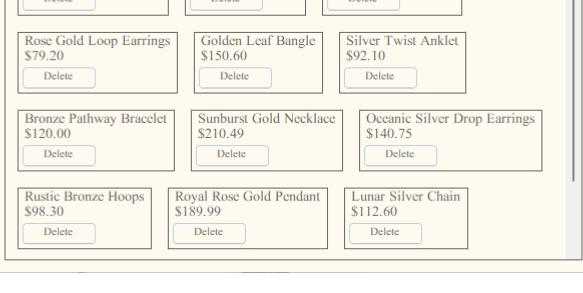
Scenario #4: "A user can sort and view the products that belong in the 'Bracelet' category."

| Step 1: User is able to click 'Bracelets' from category dropdown	Step 2: User verifies the items shown are only bracelets

Scenario #5: "An admin is able to add a product and delete a product."

Step 1: Admin clicks 'Add a Product'	Step 2: Admin fills in all the fields and the clicks 'Add Product' to get the verification message

Step 3: Admin observes the 'Silver Hoop Earrings' was added to the product list	Step 4: Admin clicks 'Delete' for the product

Step 5: Admin clicks 'Yes'	Step 6: Admin verifies 'Silver Hoop Earrings' no longer exists as a product

Strengths:

1. Dual Functionality: This application effectively supports both customer and admin users. It enables browsing, purchasing, and inventory management on the same platform.
2. User-Friendly Interface: This application presents a lot of flexibility with the options that the users have to filter their information. The filtering options (by category, metal type, popularity, and new releases) enhances customer experiences by making items simple and intuitive to discover.
3. Product Images: The use of product images even if not comprehensive, improves the look and usability of the application.
4. Inventory Control: Admins can add or delete items easily with and view key metrics like popular items and high spending customers to aid in business decisions.
5. Account Management: Users can create accounts and log in if an account already exists

Limitations:

1. Updating user information: One thing we never got to implement was that users cannot currently update their information.

2. Images are hardcoded and in the application rather than stored or referenced from the database. The images could also be more closely tailored to each item.
3. Complexity to checkout : We could have added a monetary transaction feature at check out.
4. Adjusting inventory: While we added checked out items to the orderlines table, we were not able to update all the tables and write code to handle out of stock scenarios

Aggregating Queries

Query #1: *(Get Highest Prices)*

Comment: This query shows the most expensive products in the store and how many times they have been purchased.

Implementation:

```
CREATE PROCEDURE Store.GetHighestPriceItems
AS
BEGIN
    SELECT I.ItemName, SUM(OL.Quantity) AS TotalQuantity, I.ItemPrice
    FROM Store.OrderLines OL
    INNER JOIN Store.Orders O ON O.OrderID = OL.OrderID
    INNER JOIN Store.Items I ON OL.StockItemID = I.StockItemID
    WHERE O.OrderDate <= SYSDATETIME()
    GROUP BY I.ItemName, I.ItemPrice
    ORDER BY ItemPrice DESC;
END;
```

Usage in Application:

Step 1: Admin clicks 'Highest Price' button

Results from Query: Admin views the products returned

Query #2: (*Most Popular Items*)
Comment: This query shows the top 10 items that have been purchased in the store.

Implementation:

```
CREATE PROCEDURE Store.GetPopularItems
AS
BEGIN
    SELECT TOP 10 I.ItemName, SUM(OL.Quantity) AS TotalQuantity, I.ItemPrice
    FROM Store.OrderLines OL
    INNER JOIN Store.Orders O ON O.OrderID = OL.OrderID
    INNER JOIN Store.Items I ON OL.StockItemID = I.StockItemID
```
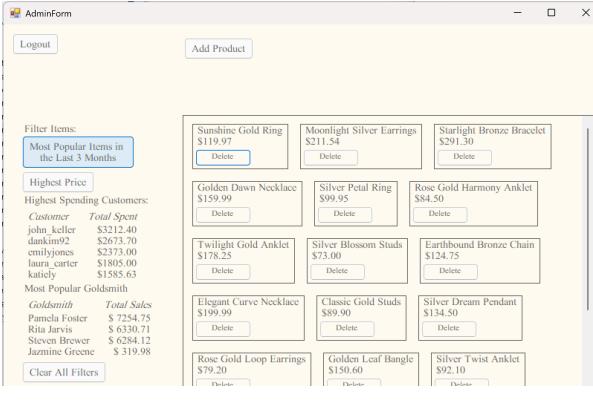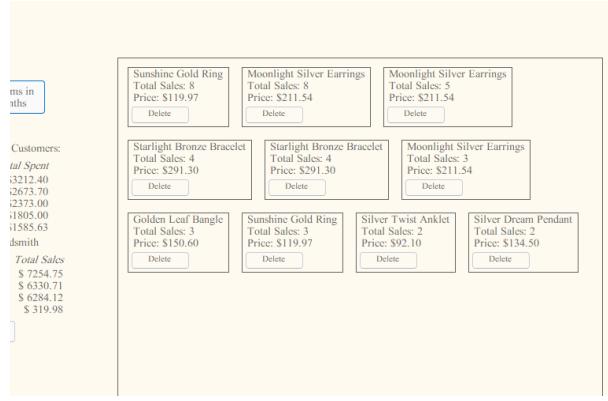
```
    WHERE O.OrderDate  >= DATEADD(MONTH, -3, SYSDATETIME())
    GROUP BY I.ItemName, I.ItemPrice, MONTH(O.OrderDate)
    ORDER BY TotalQuantity DESC;
END;
```

Usage in Application:

Step 1: Admin clicks 'Most Popular Items in the Last 3 Months' button	Results from Query: Admin views the products returned

<u>*Query #3*</u>: *(Most Popular Goldsmith)*
Comment: This query shows the top 3 goldsmiths with the highest sales.

Implementation:
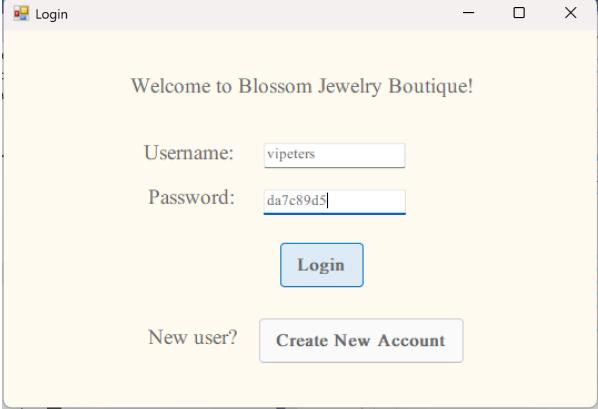
```
CREATE PROCEDURE Store.GetPopularGoldsmith
AS
BEGIN
    SELECT TOP 5 G.GoldsmithName, SUM(OL.Quantity * OL.UnitPrice) AS TotalSales
    FROM Store.Goldsmith G
    INNER JOIN Store.Items I ON I.StockItemID = G.GoldsmithID
    INNER JOIN Store.OrderLines OL ON OL.StockItemID = I.StockItemID
    GROUP BY G.GoldsmithName
    ORDER BY TotalSales DESC;
END;
```

Usage in Application:

Step 1: Admin logs into admin side	Results from Query: Admin views the most

	popular goldsmiths along with their highest sales
Login window: Welcome to Blossom Jewelry Boutique! Username: vipeters Password: da7c89d5 Login New user? Create New Account	katiely $1383.03 Most Popular Goldsmith Goldsmith Total Sales Pamela Foster $ 7254.75 Rita Jarvis $ 6330.71 Steven Brewer $ 6284.12

Query #4: (*Highest Spending customers*)

Comment: This query shows the customers that have spent the most money on the application.
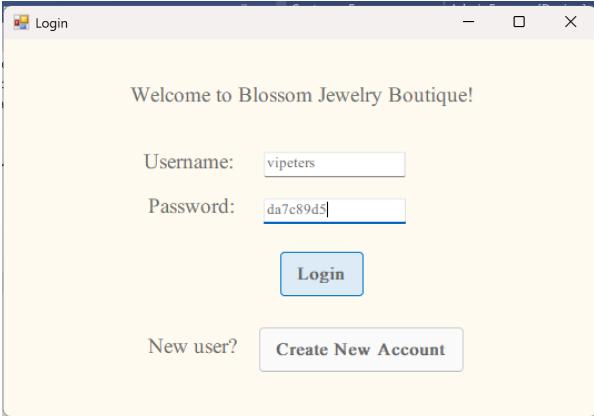
Implementation:

```
CREATE PROCEDURE Store.GetHighestSpendingCustomers
AS
BEGIN
  WITH SpendersCTE (Username, TotalSpent)
  AS(
    SELECT TOP 5 U.Username, SUM(OL.Quantity * OL.UnitPrice) AS TotalSpent
    FROM Store.[User] U
    INNER JOIN Store.Orders O ON U.UserID = O.UserID
    INNER JOIN Store.OrderLines OL ON OL.OrderID = O.OrderID
    GROUP BY U.Username
  )
  SELECT SC.Username, SC.TotalSpent,
  RANK() OVER(ORDER BY SC.TotalSpent DESC) AS [Rank]
  FROM SpendersCTE SC
    ORDER BY TotalSpent DESC;
  END;
```

Usage in Application:

Step 1: Admin logs into admin side	Results from Query: Admin views the highest spending customers along with their total spent

Summary & Discussion

Project Completeness Compared to Proposal:
Our project expanded beyond what our original proposal was.
Rather than developing a web application using JavaScript and other frameworks, we opted to create an application {on visual studio}? Given our skillset, this was an easier tool to use. We added the option for new users to create a new account, this was an addition to our initial only user login. We also decided to include pictures to represent different items to improve the user experience for the demo. These pictures were stored in the program and not in the database. While our project was originally client focused, we decided to add more features to the admin controls. These features enabled the admin to add and delete items from the inventory. This addition helped us fulfill the requirement for tables supporting Data Operations. We also changed our selection of aggregated queries because we realized the one that we had previously added did not use the GROUP BY function.

Changes Completed:
Changes to database:
Our database design went through a couple of minor changes as we saw the project develop. Firstly, we decided to remove the Quantity variable from the Goldsmith table in order to eliminate redundancy since that data was already being stored in the Store.Items table. Another change that we made was to include a new table, Store.Cart. This table was created to ease the transfer of the customer's checked out items information to be stored in the Orders and OrderLines tables.

Lessons learned:
Working on this project, we learned the importance of working with different branches when using Github. We also learned how to use SQL and Visual Studio simultaneously. This was something that none of us had previous experience with so it was interesting to experiment with. We also learned the importance of understanding the scope of the project. While coding, we kept finding more features that we hadn't initially thought of. This shows the importance of planning and writing maintainable code. During the course of the project, we found ourselves going off track instead of focusing on the requirements. This painfully taught us that we should constantly be referencing the guidelines during our working project.

Future Endeavors:
While we were able to add a few pictures for purposes of demonstration, in the future we would like to add individualized pictures for every item. We would also like to explore having the images stored in the database. With that functionality the way, the admins can add or update the pictures as they please.
We would also like to make the queries a lot more customizable. Instead of having preset filters, we would like the admins to be able to set their own presets for their desired ranges. Lastly, we would like to allow the users to edit their information in case any of it changes without them losing their information.