Database System for Music Album Online Stores

Shengwen Ding[1] and Yang Cao[2]

[1] 4594942, shd151@pitt.edu, University of Pittsburgh, PA, USA
[2] 4600098, yac70@pitt.edu, University of Pittsburgh, PA, USA

Abstract. Music is an essential part of our daily life. People listen to music and form a connection with the artists and other people. Music can also relax us from the busy schedules of study and work. The database system we build is for music lovers: an online music album store for browsing and purchasing music albums from different genres and artists. Besides the customer interface, the system also has an administrator interface for the salesperson to keep track of real-time information on the inventory amount of the albums, the transactions information, the customers information, and the analysis of the sales performance. To ensure robustness, the database system also includes error-checking functions that will send users error messages if not operate properly.

Keywords: Database System, E-Commerce, ASP.NET MVC, SQL Server.

1 Overview of the System

In this project, we design and implement a database system for music album online stores that can automate the functions of a music album sales company with E-Commerce. As is shown in the system's name. the specific product we select to sell is the music album.

The users of this system are divided into two categories: the customer and the salesperson. The customer is able to browse all the music albums, search (focused and less focused searching) for a particular item and make purchases. The salesperson can keep track of real-time updated data of inventory amounts, transactions, user information, etc., and create, edit, browse, and delete music album data. Besides, the system provides data aggregation functions, for example, the salesperson can check the current aggregate sales and profit of the music albums, the top product, etc. that reflect the transaction history and sales performance.

A robust database system should also have error-checking functions. Therefore, our system contains various application-dependent integrity constraints which are shown in section 8 of this report.

The framework and codes we mainly use are from Microsoft's ASP.NET MVC [1].

2 Assumptions

In order to simplify our database system and to focus on the core requirements, some assumptions have been made. However, the actual situation in our daily life will be much more complex, so we list some improvements to the proposed assumptions in section 9 of this report.

2.1 Assumptions on *Salesperson*

We assume that every salesperson works in only one store and will never be transferred to another store.

2.2 Assumptions on *Stores* and *Regions*

We assume that every customer's address locates in one of the regions (States) of the stores. When a customer places an order, the system will select the region according to the customer's shipping address, and then match a store in that region.

2.3 Assumptions on *Product Inventory*

We assume that every music album (product) is stored in one single inventory. Therefore, when we say the inventory amount of one music album, we mean the total amount left in the inventory, not in the stores or regions. When a customer placed an order, the album will be firstly retrieved from the inventory, then transported to the specific region and store, and lastly be shipped to the customer's address.

2.4 Assumptions on *Data*

Our database does not collect some of the data from the customers, such as income, marriage status, gender, age, etc. The reason for this is that we think the customers may not be willing to provide such private information in real situations.

3 E-R Diagram

Our database contains ten relations as shown in the E-R Diagram in Fig. 1.

1. The attributes of the **Customers** entity are Customer Id (Primary Key), User Name, First Name, Last Name, Address, City, State, Postal Code, Country, Phone, and Email.
2. The attributes of the **Albums** (the products) entity are Album Id (Primary Key), Title, and Price. Each album is categorized by one genre and is created by one artist.
3. The attributes of the **Genres** entity are Genre Id (Primary Key), Name, and Description.
4. The attributes of the **Artists** entity are Artist Id (Primary Key) and Name.
5. The attributes of the **Salespersons** entity are Salesperson Id (Primary Key), Name, Job Title, Email, and Salary. Each salesperson works in one store and not all salespersons are managers.

6. The attributes of the **Stores** entity are Store Id (Primary Key), Address, and Number of Salesperson. Each store is located in one region.

7. The attributes of the **Regions** entity are Region Id (Primary Key) and Region Name. Each region is managed by one manager.

8. The attributes of the **Orders** entity are Order Id (Primary Key), Order Date, and Total. Each order is placed by one customer, contains one or several albums with order detail information, and is processed by one salesperson.

9. The attributes of the **OrderDetails** entity are OrderDetailId (Primary Key), Quantity, and UnitPrice. Each order detail contains specific order information of one album in the order.

10. The attributes of the **Cart** entity are RecordId (Primary Key), CartId, Count, and DateCreated. Each cart is created by one customer and contains one or more albums information that the customer adds.

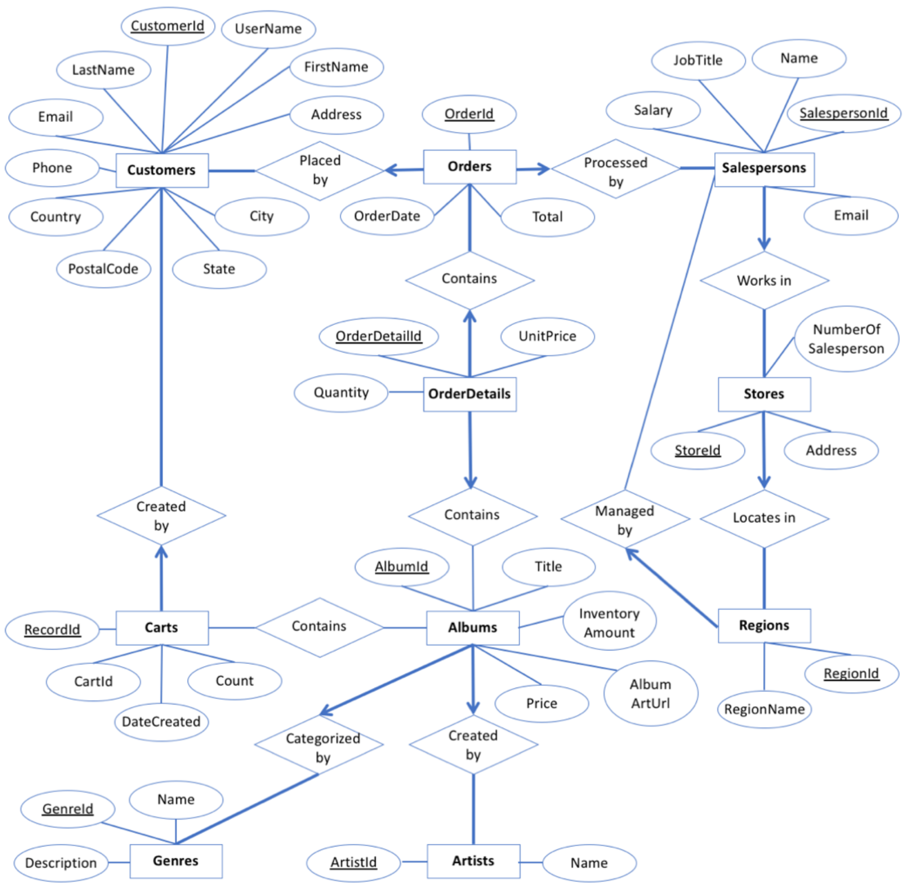


Fig. 1. E-R Diagram of the database system for music album online store

4 Relational Schema

The following is a set of relational schemas resulting from our E-R diagram. Each schema includes indexes, primary keys, etc.

Table 1. Customers

Key	Column	Type
Primary Key	CustomerId	int
	UserName	varchar
	FirstName	varchar
	LastName	varchar
	Address	varchar
	City	varchar
	State	varchar
	PostalCode	varchar
	Country	varchar
	Phone	varchar
	Email	varchar

Table 2. Albums

Key	Column	Type
Primary Key	AlbumId	int
Foreign Key	GenreId	int
Foreign Key	ArtistId	int
	Title	varchar
	Price	decimal
	InventoryAmount	int
	AlbumArtUrl	varchar

Table 3. Genres

Key	Column	Type
Primary Key	GenreId	int
	Name	varchar
	Description	varchar

Table 4. Artists

Key	Column	Type
Primary Key	ArtistId	int

	Name	varchar

Table 5. Salespersons

Key	Column	Type
Primary Key	SalespersonId	int
Foreign Key	StoreId	int
	Name	varchar
	JobTitle	varchar
	Salary	decimal
	Email	varchar

Table 6. Stores

Key	Column	Type
Primary Key	StoreId	int
Foreign Key	RegionId	int
	Address	varchar
	NumberOfSalesperson	int

Table 7. Regions

Key	Column	Type
Primary Key	RegionId	int
Foreign Key	SalespersonId	int
	RegionName	varchar

Table 8. Orders

Key	Column	Type
Primary Key	OrderId	int
Foreign Key	CustomerId	int
Foreign Key	SalespersonId	int
	OrderDate	datetime
	Total	numeric

Table 9. OrderDetails

Key	Column	Type
Primary Key	OrderDetailId	int
Foreign Key	OrderId	int
Foreign Key	AlbumId	int
	UnitPrice	decimal

	Quantity	int

Table 10. Carts

Key	Column	Type
Primary Key	RecordId	int
Foreign Key	CustomerId	int
Foreign Key	AlbumId	int
	CartId	int
	Count	int
	DateCreated	datetime

5 DDL Statement

The following are the DDL statements that create the relational schema.

5.1 Create *Customers* Table

```
CREATE TABLE ‘Customers’ (
        ‘CustomerId’ INT (45) NOT NULL,
        ‘UserName’ VARCHAR (45),
        ‘FirstName’ VARCHAR (45) NOT NULL,
        ‘LastName’ VARCHAR (45) NOT NULL,
        ‘Address’ VARCHAR (255) NOT NULL,
        ‘City’ VARCHAR (45) NOT NULL,
        ‘State’ VARCHAR (45) NOT NULL,
        ‘PostalCode’ VARCHAR (45),
        ‘Country’ VARCHAR (45),
        ‘Phone’ VARCHAR (45),
        ‘Email’ VARCHAR (45),
        PRIMARY KEY (‘CustomerId’)
);
```

5.2 Create *Albums* Table

```
CREATE TABLE ‘Albums’ (
        ‘AlbumId’ INT (45) NOT NULL,
        ‘GenreId’ INT (45) NOT NULL,
        ‘ArtistId’ INT (45) NOT NULL,
        ‘Title’ VARCHAR (225) NOT NULL,
        ‘Price’ DECIMAL (10, 2) NOT NULL,
        ‘InventoryAmount’ INT (MAX) NOT NULL,
        ‘AlbumArtUrl’ VARCHAR (225),
        PRIMARY KEY (‘AlbumId’),
        FOREIGH KEY (‘Genres’. ‘GenreId’, ‘Artists’. ‘ArtistId’)
);
```

5.3 Create *Genres* Table

```
CREATE TABLE ‘Genres’ (
        ‘GenreId’ INT (45) NOT NULL,
        ‘Name’ VARCHAR (225) NOT NULL,
        ‘Description’ VARCHAR (225),
        PRIMARY KEY (‘GenreId’),
);
```

5.4 Create *Artists* Table

```
CREATE TABLE ‘Artists’ (
        ‘ArtistId’ INT (45) NOT NULL,
        ‘Name’ VARCHAR (225) NOT NULL,
        PRIMARY KEY (‘ArtistId’),
);
```

5.5 Create *Salespersons* Table

```
CREATE TABLE ‘Salespersons’ (
        ‘SalespersonId’ INT (45) NOT NULL,
        ‘StoreId’ INT (45) NOT NULL,
        ‘Name’ VARCHAR (225) NOT NULL,
        ‘Salary’ DECIMAL (10, 2) NOT NULL,
        ‘JobTitle’ VARCHAR (225),
        ‘Email’ VARCHAR (225),
        PRIMARY KEY (‘SalespersonId’),
        FOREIGH KEY (‘Stores’. ‘StoreId’)
);
```

5.6 Create *Stores* Table

```
CREATE TABLE ‘Stores’ (
        ‘StoreId’ INT (45) NOT NULL,
        ‘RegionId’ INT (45) NOT NULL,
        ‘Address’ VARCHAR (255) NOT NULL,
        ‘NumberOfSalesperson’ INT (MAX) NOT NULL,
        PRIMARY KEY (‘StoreId’),
        FOREIGH KEY (‘Regions’. ‘RegionId’)
);
```

5.7 Create *Regions* Table

```
CREATE TABLE ‘Regions’ (
        ‘RegionId’ INT (45) NOT NULL,
        ‘SalespersonId’ INT (45),
        ‘RegionName’ VARCHAR (225) NOT NULL,
        PRIMARY KEY (‘RegionId’),
        FOREIGH KEY (‘Salespersons’. ‘SalespersonId’)
);
```

5.8 Create *Orders* Table

```
CREATE TABLE ‘Orders’ (
        ‘OrderId’ INT (45) NOT NULL,
        ‘CustomerId’ INT (45) NOT NULL,
        ‘SalespersonId’ INT (45) NOT NULL,
        ‘OrderDate’ DATETIME NOT NULL,
        ‘Total’ NUMERIC (10, 2) NOT NULL,
        PRIMARY KEY (‘OrderId’),
        FOREIGH KEY (‘Customers’. ‘CustomerId’, ‘Salespersons’.
‘SalespersonId’)
);
```

5.9 Create *OrderDetails* Table

```
CREATE TABLE ‘OrderDetails’ (
        ‘OrderDetailId’ INT (45) NOT NULL,
        ‘OrderId’ INT (45) NOT NULL,
        ‘AlbumId’ INT (45) NOT NULL,
        ‘UnitPrice’ DECIMAL (10, 2) NOT NULL,
        ‘Quantity’ INT (45) NOT NULL,
        PRIMARY KEY (‘OrderDetailId’),
        FOREIGH KEY (‘Orders’. ‘OrderId’, ‘Albums’. ‘AlbumId’)
);
```

5.10 Create *Carts* Table

```
CREATE TABLE ‘Carts’ (
        ‘RecordId’ INT (45) NOT NULL,
        ‘CustomerId’ INT (45) NOT NULL,
        ‘AlbumId’ INT (45) NOT NULL,
        ‘Count’ INT (45) NOT NULL,
        ‘AlbumId’ INT (MAX) NOT NULL,
        ‘DateCreated’ DATETIME NOT NULL,
        PRIMARY KEY (‘RecordId’),
        FOREIGH KEY (‘Customers’. ‘CustomerId’, ‘Albums’. ‘AlbumId’)
);
```

6 Front-end Design and Front-end to Back-end Connection

The framework we use for this project is the ASP.NET MVC provided by Microsoft [1]. Our code is mainly based on this tutorial, and we add some new functions such as the searching of the products, data aggregation, etc. This architectural pattern is easy to use for web-design beginners and has a good connection between the front-end, the back-end, and the database. Its structure can be separated into three main parts Fig. 2.: the model, the view, and the controller, which are shortened for “M”, “V”, and “C” in the framework’s name.

The model contains the business logic of the system. Developers can retrieve and store the model states in a database and implement the logic for the data domain. For example, if a salesperson wants to update the information of a product, the model retrieves the data from the database, operates on it, and then updates the information back to the table where the product data is stored in the SQL server.

The view contains the user interface (UI) logic of the system. It is generated from the model and displays the information that users can see on the screen, for instance, a text box or a drop-down list.

The controller contains the input logic of the system. It works with the model and the view. In general, it reacts to the user’s actions, selects a view to display UI, passes the data to the model, and then queries the database.

The MVC has several advantages. Firstly, it is lightweight, highly testable, and easy to use for beginners. Besides, the separation among the three components, the model, the view, and the controller, makes it efficient and effective for parallel development among various developers. Last but not the least, it is integrated with the existing features of ASP.NET and is familiar to many experienced web designers, which means it has a high level of recognition.

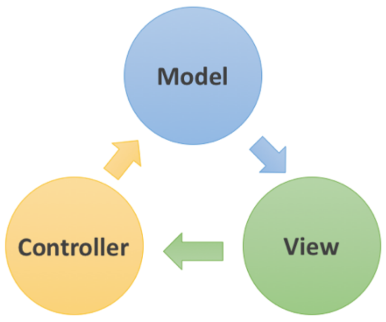


Fig. 2. The three main parts of the MVC framework: the Model, the View, and the Controller.

7 Example Screen Shots

The following are some example screen shots from our database system for music album online stores.

7.1 Customer

Fig. 3. is the Home Page where a customer can browse the music albums for sale, search for a target album in the searching box at the top, see the most recently ordered albums at the bottom, or select a genre of the albums on the left menu bar.

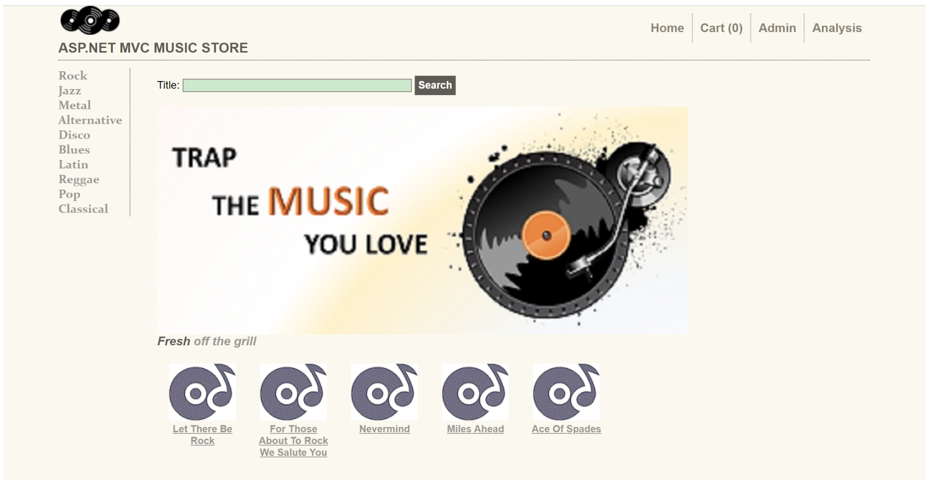
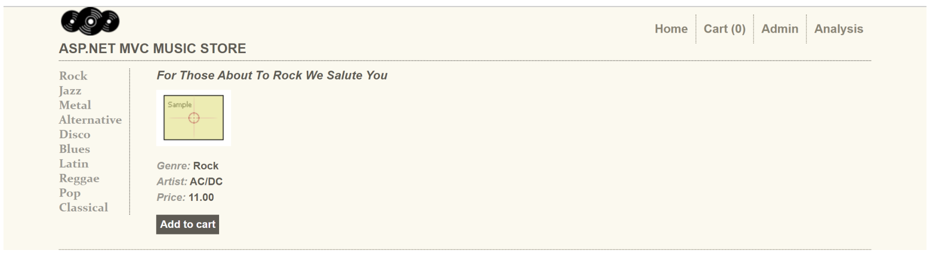


Fig. 3. Home Page.

After selecting an album, the customer can see the title, the genre, the artist and the price of the album on the Album Information Page Fig. 4. By clicking the “Add to cart” button, the album will be added to the cart.

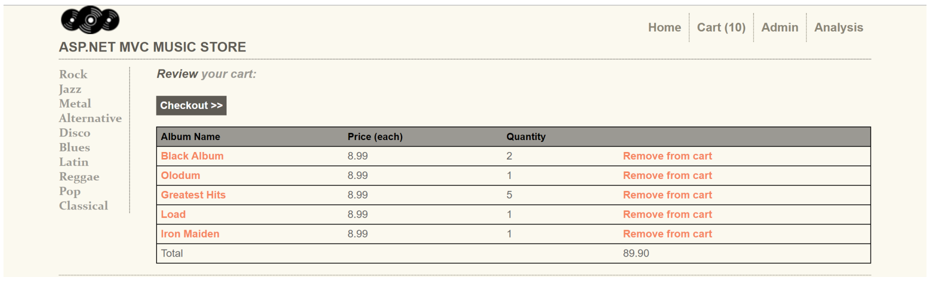


For Those About To Rock We Salute You

Genre: Rock
Artist: AC/DC
Price: 11.00

Fig. 4. Album Information Page.

Fig. 5. is a demo of a Cart Review Page. In this page, the customer can view the information of the albums added to the cart and change the quantities of them. Then, by clicking the “Checkout” button, the customer will go to Check Out Page and will be required to provide shipping information such as name, address, state, phone, etc.

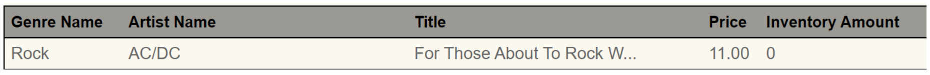


Review *your cart:*

Album Name	Price (each)	Quantity	
Black Album	8.99	2	Remove from cart
Olodum	8.99	1	Remove from cart
Greatest Hits	8.99	5	Remove from cart
Load	8.99	1	Remove from cart
Iron Maiden	8.99	1	Remove from cart
Total			89.90

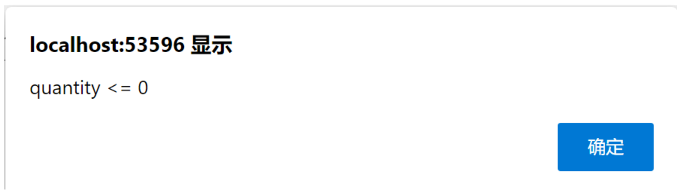
Fig. 5. Cart Review Page

In case one album is sold out, which means the inventory amount of this album equals zero, the system will give the customer an alert and the album cannot be added to the cart. For example, if the album in Fig. 6.in is sold out, the customer will notice a message like Fig. 7. on the screen. In section 9, we list another two ways to improve the solution of this case which we believe can provide better user experience.



Genre Name	Artist Name	Title	Price	Inventory Amount
Rock	AC/DC	For Those About To Rock W...	11.00	0

Fig. 6. The inventory amount of a selected music album equals zero.



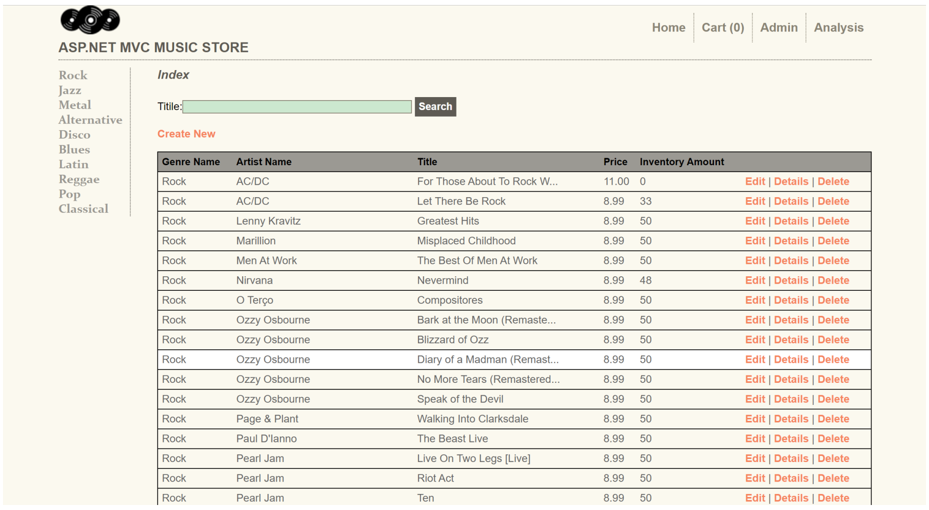
localhost:53596 显示

quantity <= 0

Fig. 7. Message occurs when the inventory amount of a selected album is zero.

7.2 Salesperson

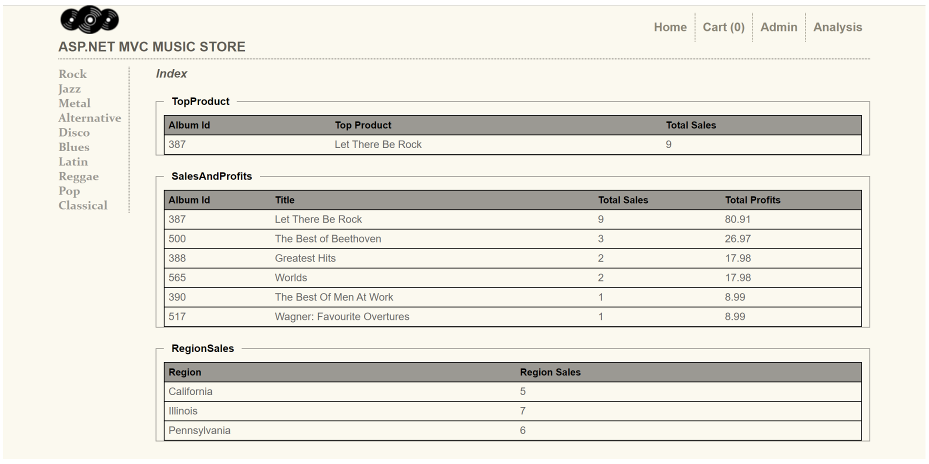
Fig. 8. is the Album List Page where a salesperson can browse the music albums for sale, search for a target album in the searching box at the top, create new albums, edit or see the details of existing albums, or delete an album.



Genre Name	Artist Name	Title	Price	Inventory Amount	
Rock	AC/DC	For Those About To Rock W...	11.00	0	Edit | Details | Delete
Rock	AC/DC	Let There Be Rock	8.99	33	Edit | Details | Delete
Rock	Lenny Kravitz	Greatest Hits	8.99	50	Edit | Details | Delete
Rock	Marillion	Misplaced Childhood	8.99	50	Edit | Details | Delete
Rock	Men At Work	The Best Of Men At Work	8.99	50	Edit | Details | Delete
Rock	Nirvana	Nevermind	8.99	48	Edit | Details | Delete
Rock	O Terço	Compositores	8.99	50	Edit | Details | Delete
Rock	Ozzy Osbourne	Bark at the Moon (Remaste...	8.99	50	Edit | Details | Delete
Rock	Ozzy Osbourne	Blizzard of Ozz	8.99	50	Edit | Details | Delete
Rock	Ozzy Osbourne	Diary of a Madman (Remast...	8.99	50	Edit | Details | Delete
Rock	Ozzy Osbourne	No More Tears (Remastered...	8.99	50	Edit | Details | Delete
Rock	Ozzy Osbourne	Speak of the Devil	8.99	50	Edit | Details | Delete
Rock	Page & Plant	Walking Into Clarksdale	8.99	50	Edit | Details | Delete
Rock	Paul D'Ianno	The Beast Live	8.99	50	Edit | Details | Delete
Rock	Pearl Jam	Live On Two Legs [Live]	8.99	50	Edit | Details | Delete
Rock	Pearl Jam	Riot Act	8.99	50	Edit | Details | Delete
Rock	Pearl Jam	Ten	8.99	50	Edit | Details | Delete

Fig. 8. Album List Page.

Fig. 9. is the Data Aggregation Page (Analysis Page) where the salesperson can check the aggregation information: the top product with the best sales, the sales and profits of all products, the comparison of sales of businesses in different regions.



TopProduct

Album Id	Top Product	Total Sales
387	Let There Be Rock	9

SalesAndProfits

Album Id	Title	Total Sales	Total Profits
387	Let There Be Rock	9	80.91
500	The Best of Beethoven	3	26.97
388	Greatest Hits	2	17.98
565	Worlds	2	17.98
390	The Best Of Men At Work	1	8.99
517	Wagner: Favourite Overtures	1	8.99

RegionSales

Region	Region Sales
California	5
Illinois	7
Pennsylvania	6

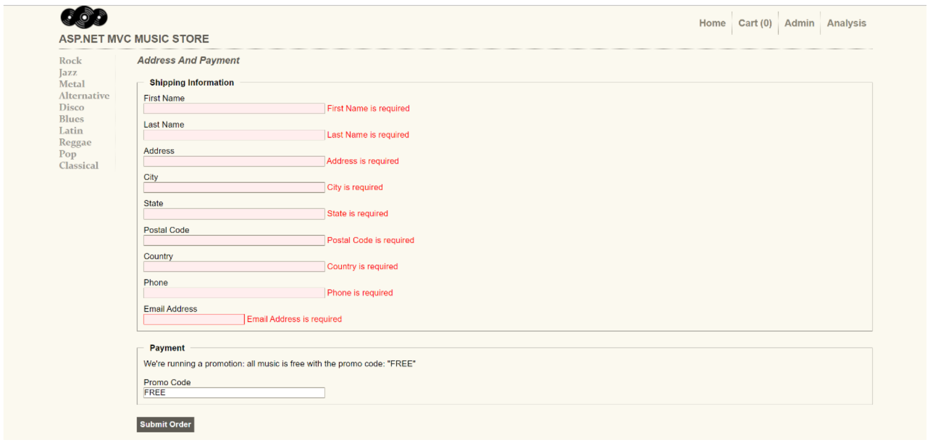
Fig. 9. Data Aggregation Page.

8 Testing Efforts and Erroneous Cases

In our database system, there are error-checking functions in both customer and salesperson interfaces.

8.1 Customer

Submit an Order. Before submitting an order, the customer needs to fill the shipping information. The "First Name", "Last Name", "Address", "City", "State", "Postal Code", "Country", "Phone", and "Email Address" must be filled with the right type. If any of the data is left blank, the customer will see an error message, as is shown in Fig. 10. and the order cannot be submitted until all the data are correctly filled.



Address And Payment

Shipping Information

First Name
First Name is required

Last Name
Last Name is required

Address
Address is required

City
City is required

State
State is required

Postal Code
Postal Code is required

Country
Country is required

Phone
Phone is required

Email Address
Email Address is required

Payment

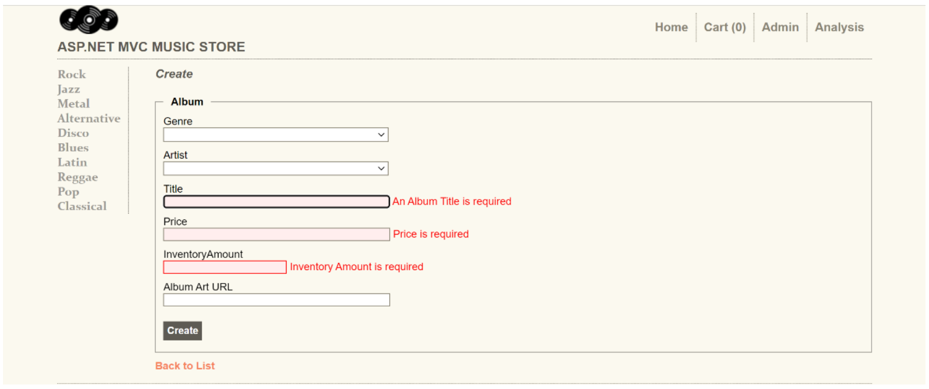
We're running a promotion: all music is free with the promo code: "FREE"

Promo Code
FREE

Fig. 10. If a customer leaves the required data blank while submitting an order, error messages will occur.

8.2 Salesperson

Create an Album. When creating an album, the “Title”, “Price”, and “Inventory Amount” must be filled with the right type and within the correct range. If any of the data is left blank or out of range, the salesperson will see an error message, as is shown in Fig. 11. And Fig. 12. and the album cannot be created until all the data are correct.

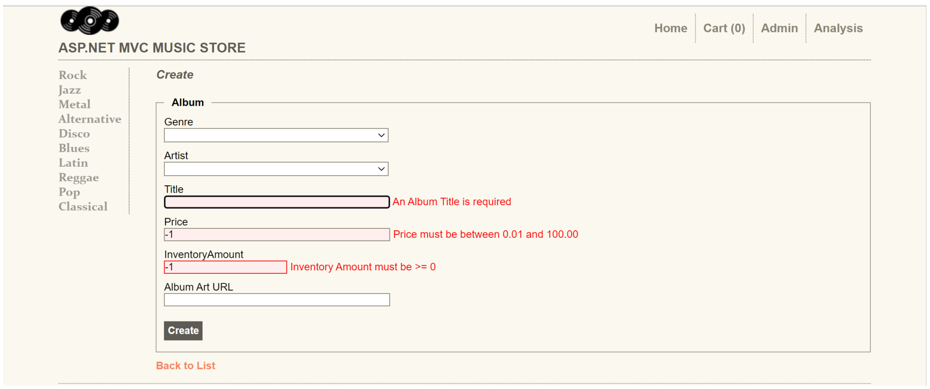


An Album Title is required

Price is required

Inventory Amount is required

Fig. 11. If a salesperson leaves the required data blank while creating an album, error messages will occur.



Price: -1

InventoryAmount: -1

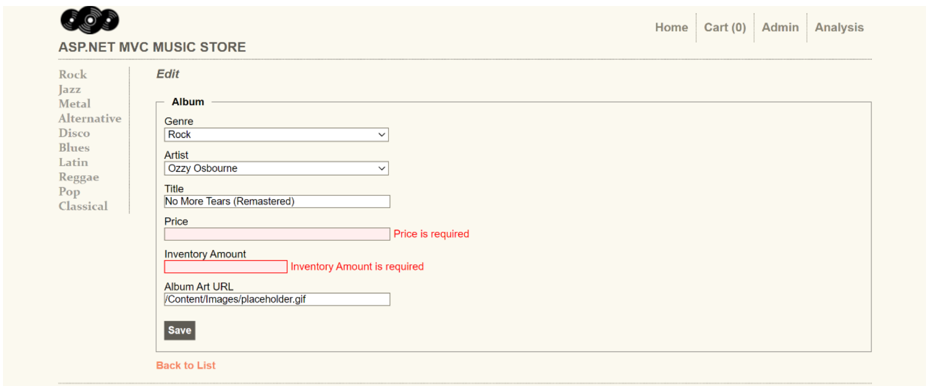
An Album Title is required

Price must be between 0.01 and 100.00

Inventory Amount must be >= 0

Fig. 12. If the data the salesperson fills is out of range when creating an album, error messages will occur.

Edit an Album. Similarly, when editing an album, the “Title”, “Price”, and “Inventory Amount” must also be filled with the right type and within the correct range. If any of the data is left blank or out of range, the salesperson will see an error message, as is shown in Fig. 13. And Fig. 14. and the album information cannot be saved until all the data are correct.



Edit

Album

Genre
Rock

Artist
Ozzy Osbourne

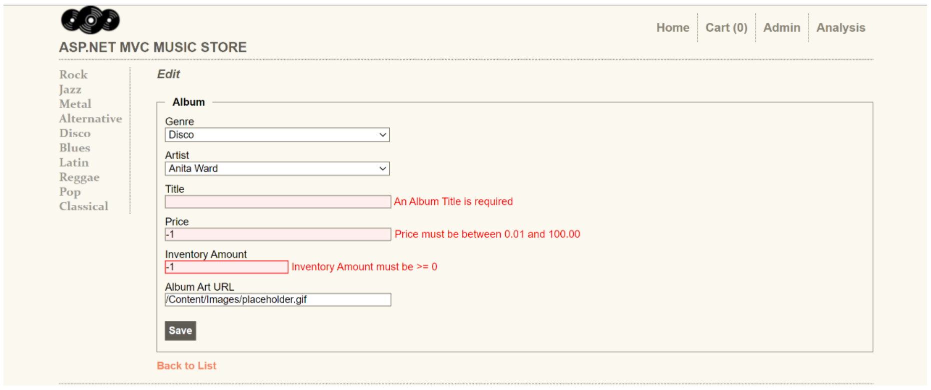
Title
No More Tears (Remastered)

Price
Price is required

Inventory Amount
Inventory Amount is required

Album Art URL
/Content/Images/placeholder.gif

Fig. 13. If a salesperson leaves the required data blank while editing an album, error messages will occur.



Edit

Album

Genre
Disco

Artist
Anita Ward

Title
An Album Title is required

Price
-1
Price must be between 0.01 and 100.00

Inventory Amount
-1
Inventory Amount must be >= 0

Album Art URL
/Content/Images/placeholder.gif

Fig. 14. If the data the salesperson fills is out of range when editing an album, error messages will occur.

9 System’s Limitations and the Possibilities for Improvements

9.1 Limitations and Improvements in *Salesperson*

In the assumption, we assume that each salesperson stays in only one store and will not relocate to another. However, in reality, the salesperson may work in a store for a period, leave the store and then join another store in the same region or even in a new region. Therefore, we are going to add a status attribute in the salesperson table in the database to reflect whether the salesperson works or has worked in one store to achieve a better reflection of reality.

9.2 Limitations and Improvements in *Stores* and *Regions*

In the assumption, we assume that each customer locates in one region that is listed in the region table in the database. This means that the system can find a matching region to ship the products. Nevertheless, there may be some customers whose shipping address is out of scope. For instance, our shops are located in every state in the US, not in other countries. A customer from Toronto, Canada placed an order, but no store in the US matches. To deal with this problem in the future, we are going to improve our system by matching the stores in the nearest region with the shipping address, not the region where the shipping address locates. In this way, the sales range of our products becomes even wider.

9.3 Limitations and Improvements in *Product Inventory*

In the assumption, we assume that all the products are stored in one single inventory to simplify the system. But the reality is that different stores will also have a certain number of products. For improvement, we decide to add an inventory table to the database to tell the number of each product and where it stores.

Besides, adding a page-turning function to the product listing pages will make it easier for users to operate our system.

Last but not the least, we think of three ways to deal with the case when the inventory number equals zero：

1. If the inventory amount of a product equals zero, this product cannot be added to the cart and an error message occurs (the method we use in this system);
2. Or if the inventory amount of a product equals zero, this product can be added to the cart, but the customer will see a “zero inventory” alert;
3. Or to be more complex, no matter whether the inventory amount of a certain product equals zero or not, the product can always be added to the cart. The customer will see a “zero inventory” alert if there is none left, and the alert will disappear if the product is restocked. Moreover, in some special cases such as Black Friday sales, there might be some products left when the customer adds them to the cart, but these products are sold out by the time the customer wants to make a purchase. Therefore, the database system needs to compare the inventory amount at each checkout.

9.4 Limitations and Improvements in *Login Function*[1]

Currently, in our system, the users do not need to log in to their accounts. However, to distinguish the customer and salesperson users, the login function should be included in the system in the future. By then, the customer will only see the "Home", "Album List" and "Cart" pages, and have no access to the "Admin" or "Analysis" pages which are exclusive to the salesperson. The customer needs to log in to their account before making an order.

Besides, after we have the login functions, the customer should be able to browse his/her history orders if logging in to the account.

Lastly, the salesperson can see the history orders of all the customers. To ensure privacy of the customers, the customers' private information needs to be encrypted, such as real name, phone number, address, etc.

10 Acknowledgements

This project was the culmination of months of hard work, and although it wasn't perfect, it was a big step forward for us as neither of us had a bachelor's degree in computer science (One of us majored in Mathematics, and the other majored in Materials Science). So much thanks to Professor Vladimir Zadorozhny who gave as precious advices on our E-R Diagram design. And we are also grateful to our teaching assistants Jason for his continued patient help.

References

1. ASP.NET MVC Overview, Microsoft, https://learn.microsoft.com/en-us/aspnet/mvc/, last accessed 2022/11/30.

[1] The second and third paragraphs of section 9.4 are additional content based on feedback from the TA after our demonstration.