

Contents

[System Report 1](#_Toc323827320)

[Database Diagram 1](#_Toc323827321)

[Installation 2](#_Toc323827322)

[Assumptions/ Design Considerations 2](#_Toc323827323)

[Project Structure 3](#_Toc323827324)

[Login 4](#_Toc323827325)

[Navigation 5](#_Toc323827326)

[Members 5](#_Toc323827327)

[View Member 6](#_Toc323827328)

[Add Member 8](#_Toc323827329)

[Transactions 9](#_Toc323827330)

[View Transaction 10](#_Toc323827331)

[Add Transaction 12](#_Toc323827332)

[DVDs 13](#_Toc323827333)

[View DVD 14](#_Toc323827334)

[Add DVD 16](#_Toc323827335)

# System Report

## Database Diagram



## Installation

The application was designed using Visual Studio 2010; which can be opened and runnable through the developer environment. The database however will need to be installed; since it was chosen to use Microsoft SQL Server the database is external to the application.

It can be done using two options.

1. Inside the project directory, there is a file called “SQLScript.sql”. This can be executed within Microsoft SQL Server Management Studio. It will create the DVDEzy database, add the tables, and also insert default dummy data.
2. The .mdf and .ldf files were also included, so it means they can be attached to SQL Server. It will however have to be moved to an accessible folder on C drive. The folder will need to have **full control** for it to work. So to change this you will have to right click the folder, go to properties, then to security, click on “users”, and then press “edit”. The full control option can then be assigned.

Providing the folder has full control it then can be attached by right clicking on “Databases” within SQL Management Studio and clicking the “attach” option. It will then bring up a new window with the button to “Add” a database file.

After the database has been added, the connection string in the application may need to be modified. The connection string is within the “Web.config” file, up the top under <appSettings>.

The content of the connection string is:

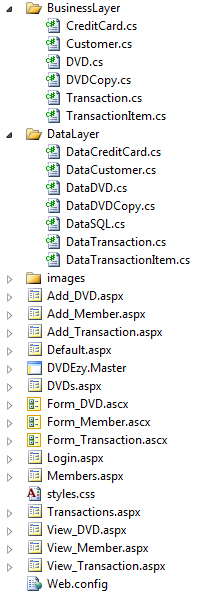
<add key="connection" value="Data Source=(local)\SQLEXPRESS;Persist Security Info=True;uid=sa;pwd=newcastle;Initial Catalog=DVDEzy;"/>

Depending on the SQL instance, the name “SQLEXPRESS” may have to be changed. By default the login details will be using the “sa” account, with the password as “newcastle”.

## Assumptions/ Design Considerations

* The design that was chosen in the system is to have a DVD contain Copy information. This design change enables the system to track which physical copies are being rented/ purchased, and whether the copy is actually in the store. It is assumed the company will have a barcode system on the physical DVDs, so each copy will also contain the barcode number to identify each record.
* The customer stores credit cards to help speed up the transaction time, and to minimize data repetition. It enables for the checkout person to simply match the card numbers. It is assumed the card type (Visa, MasterCard) doesn’t need to be stored in the card object, since that specific data is stored with the transaction.

## Project Structure

The web application is designed using 3-tier architecture layers; consisting of a business layer, a data layer, and a client layer. The business layer contains classes which interact with the data layer in order to fetch and retrieve the stored data.

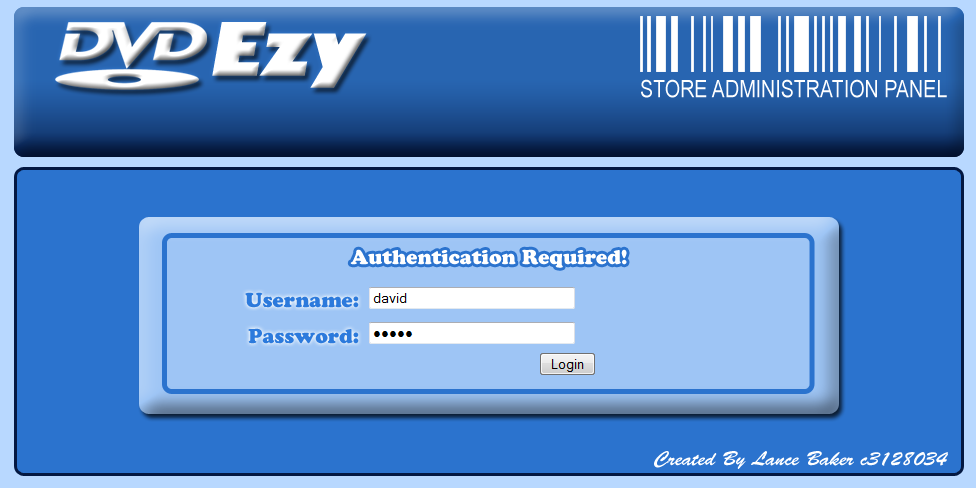
The data layer consists of a series of classes that correspond with the business layer. Each class includes the SQL statements defined as constant values, and has a series of retrieval and manipulation methods; which then interacts with a generic database helper class called DataSQL. The DataSQL class then establishes a connection and directly calls the internal SQL methods for executing the queries. The DataSQL class was also designed using Reflection, which enables for it in the retrieval methods to return instance objects (based on the generic type). So it will automatically return an instance of the data type, using the result from the query as the parameters for the constructor.

The client is designed to make use of a Master page file, which contains the basic layout of the site (including the header, and footer) and a content placeholder for the other pages. The other pages then extend off the Master page, filling in the content placeholder with desired data. The client itself will only interact with the business layer, and will not directly access the data layer.

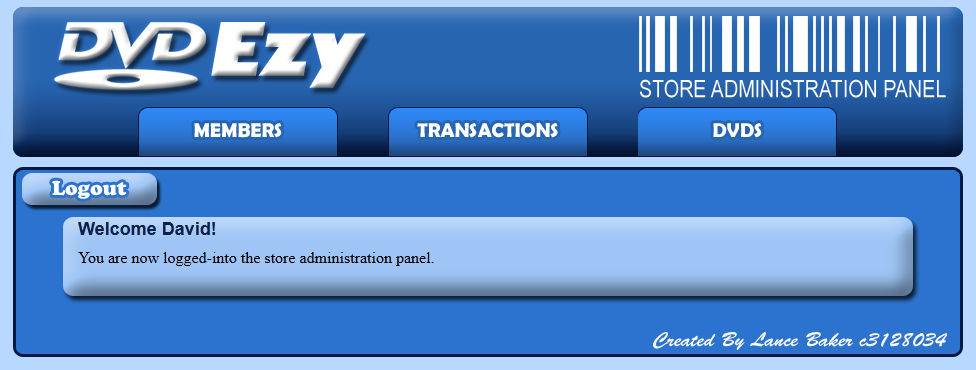
The application makes use of user controls for the forms; which allows the forms to be reused for both displaying and adding the data. The main functionality and coding is included within those controls, facilitating as much reuse as possible.

## Login

The login page is displayed automatically on start-up. If the user enters valid credentials, then it will redirect them to the starting page, show the main navigation, and enable them the ability to access other pages. In the event that a page is accessed without first logging-in, it will just redirect the user back to the login page. The logout button will destroy the session.

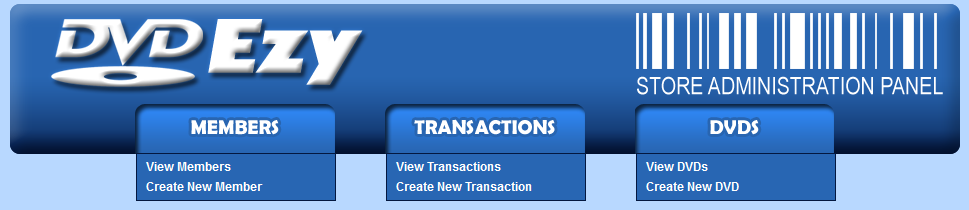


Once the user enters valid credentials it will show the navigation, and allow them to access other pages (without being redirected back to the login page).



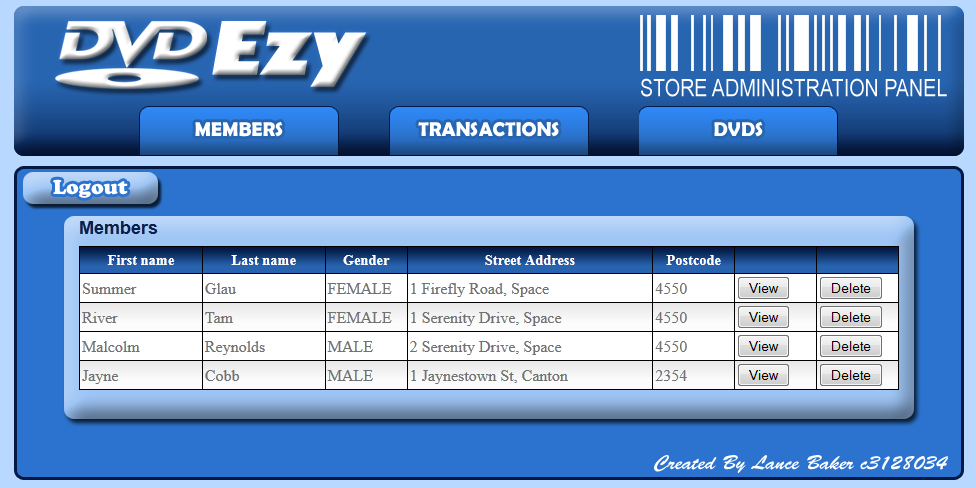
## Navigation

The navigation uses CSS styles to create the menu hovering effect. It shows the corresponding sub menu once a user has hovered over the desired button with the cursor; allowing them enough time to click on links within the newly appeared menu beneath.



## Members

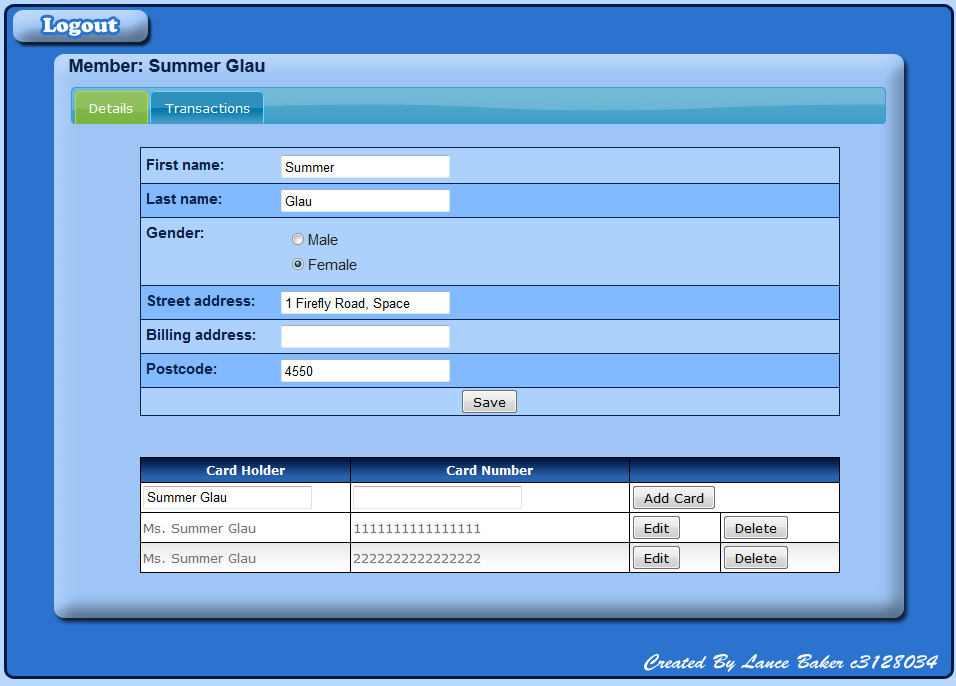
The members’ page can be accessed from the menu under “View Members”. It lists the customers that are currently stored in the system. It uses a DataGridView which is bound onto the customers’ object list. It is comprised of functionality to view and delete a record. Viewing a record will redirect the user to another page to view a specific customer object.



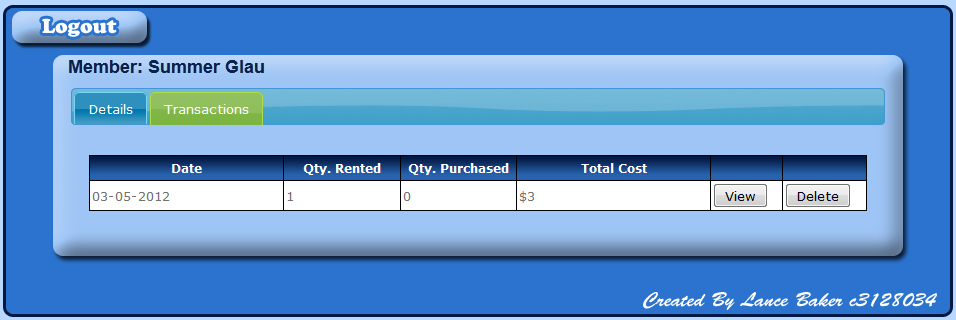
### View Member

The view member page is accessed from the DataGridView on the members’ page. It is used to show a customer object. It receives an identifier which relates to the key location in the underlying data structure, and shows that retrieved object in the form fields.

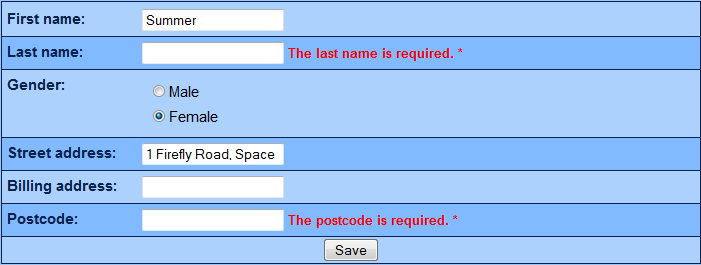
The credit cards relating to the customer object is also shown in a DataGridView; comprising of the functionality to add a new credit card, edit an existing one, and also to delete one altogether. The card holder textbox for adding a new card is automatically shown with the customer’s name which just aids in the creation of a new record. The card holder is required, since a person may use a credit card that is not directly in their name (such as a spouse or parent).



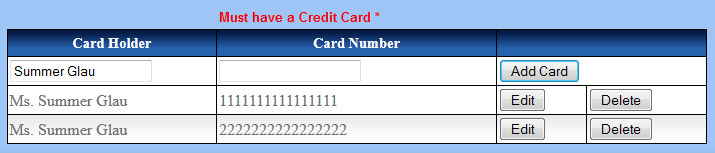
The transactions relating to the customer can also be viewed by pressing on the “Transactions” tab. It then displays a DataGridView of all the transactions made by that customer.



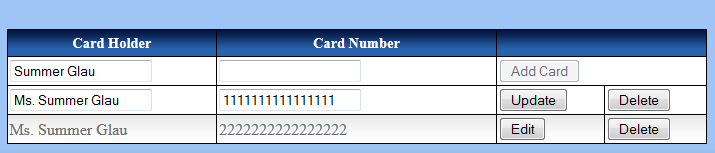
The form consists of validation specifying required fields. Also, it only allows alphabetic characters to be used in the first and last name. The billing address is the only optional field, so there isn’t any required validation. The postcode is required and will only accept a four digit number.



The credit cards relating to a customer is displayed in a DataGridView, which comprises of the ability to Insert, Edit, and Delete records. The “Add Card” button validates the input fields, and will display a message directly above the column relating to the rule. The card number is required, and will only accept a sixteen digit number (since it is the amount of digits a Visa or MasterCard will have). The card holder is also a required field. Once the “Add Card” has been pressed, the new object will be instantiated and added to the internal list. The DataGridView is then refreshed to reflect the changes made.

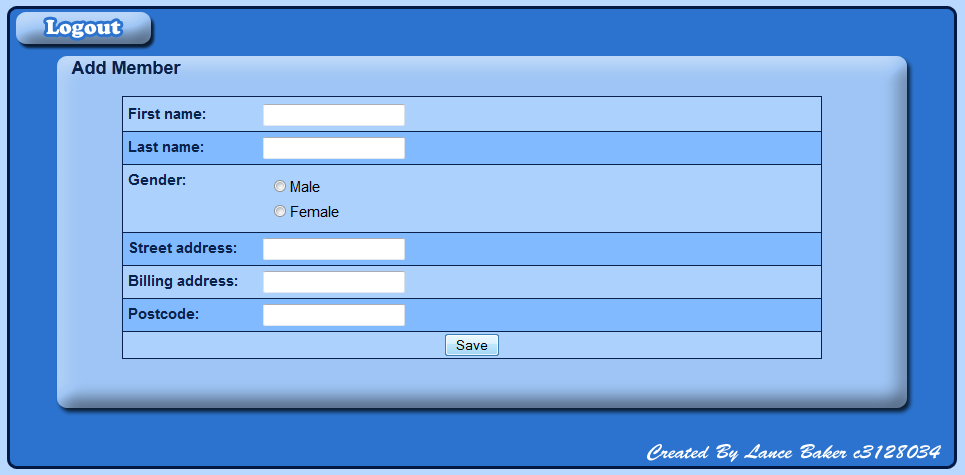


A credit card details can be changed by pressing on the Edit button; which then will disable the “Add Card” button until the Update button is pressed to make the changes to the object. If invalid data is entered; then an error message is shown on-top of the form once the Update button is pressed, it will stay in update mode allowing the user to then correct the changes. There isn’t any inline validation on the Edit record ability, since it cannot be applied to DataGridView fields. The validation rules for Editing a record is the same as Inserting; checking the fields to ensure they are filled in, and also checking the card number to ensure its sixteen digits.

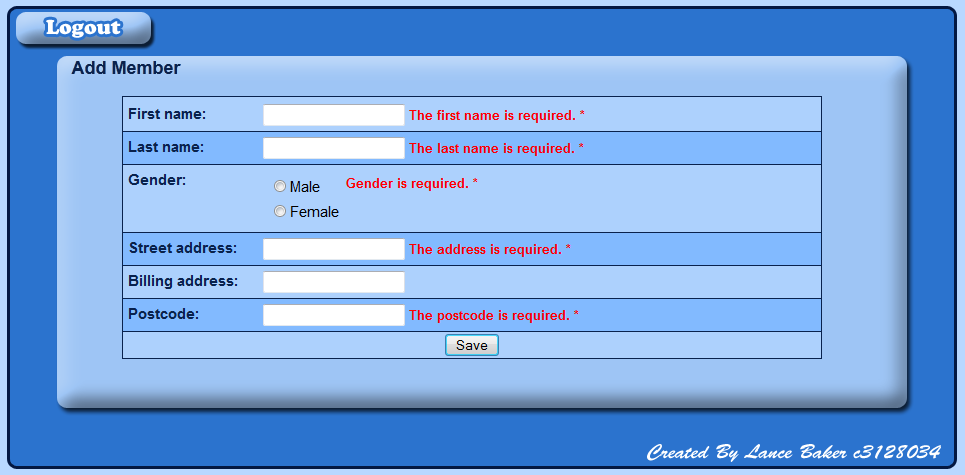


### Add Member

The add member page can be accessed from the menu under “Create New Member”. It will use the same form user control that is shown when displaying a record, except the customer object is not displayed in the form. The save method accommodates this, and will instantiate a new object instead of retrieving it from the list. It will then add the new object to the list. Once the object is added, the page will refresh, which will allow further editing of the object. It will then also show the credit cards DataGridView, allowing the ability to insert credit cards on the newly created object.



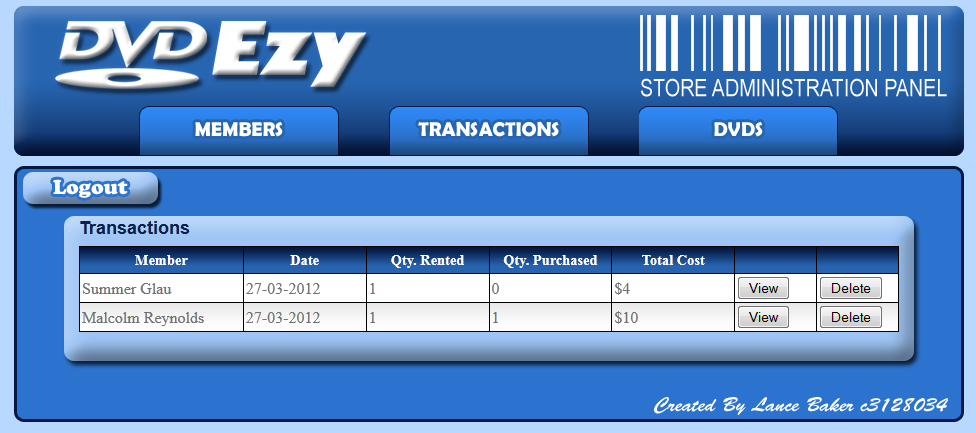
The validation on the form is the same as outlined in view member. All fields are required, with the first and last name having a restriction on only accepting alphabetic characters. The postcode can only be a four digit number.



## Transactions

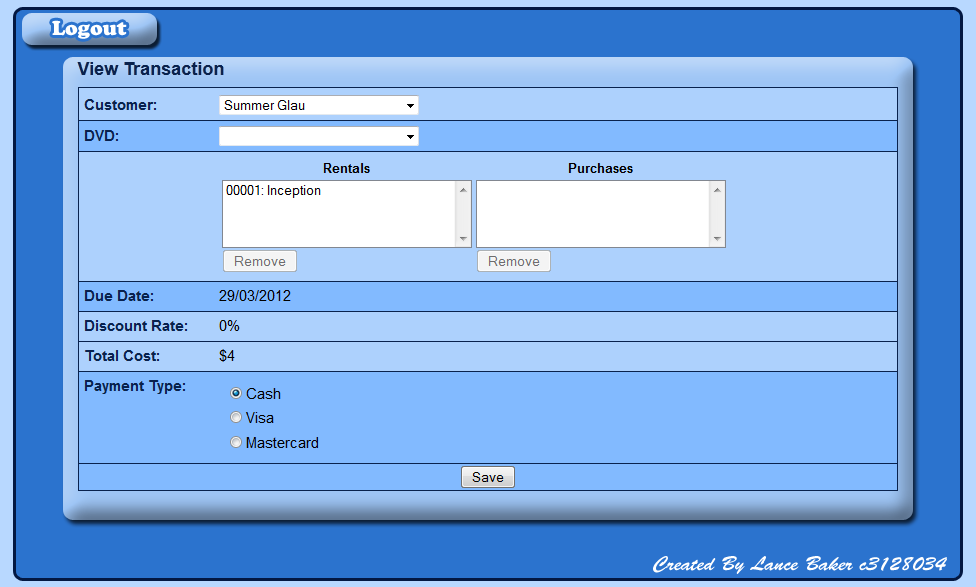
The transactions’ page can be accessed from the menu under “View Transactions”. It lists the transactions that are currently stored in the system. It uses a DataGridView which is bound onto the transactions’ object list. It is comprised of functionality to view and delete a record. Viewing a record will redirect the user to another page to view a specific transaction object.

The delete option will remove a transaction, but also will reset each containing DVD copy’s availability field to true, indicating that the copy is back in the store and ready for reuse.

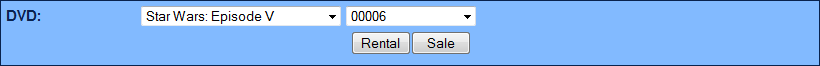


### View Transaction

The view transaction page is accessed from the parent DataGridView in transactions’ page by clicking on the View button. It is passed a unique identifier allowing the transaction to be retrieved from the internally stored list. Once it has the object, it is then passed into a method on the form user control to show the data on the form. The transaction can be completely mutated without restriction when it has been created. The due date, discount rate, and total cost are calculated properties and will depend of the contents on the transaction. If there aren’t any rentals, then it will just simply not show the due date.

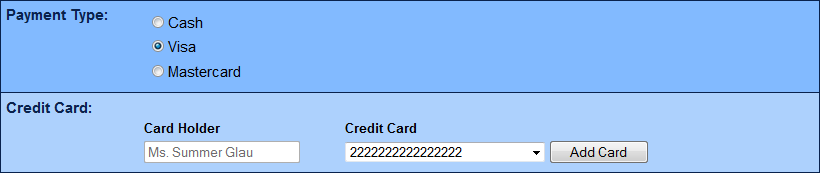


The DVD order information is added to the form with first selecting the DVD title. It will then show another dropdown box containing the copies with the ability to add it to the transaction as either a rental or a sale. A DVD will have many copies, with each copy having a barcode (that will physically be on the DVD) and also another field being the availability as to whether it is in the store. Once the copy is added to the transaction it will disappear from the dropdown list and appear in the corresponding list box. The calculated properties (due date, discount rate, and total cost) will then be updated to reflect the changes.

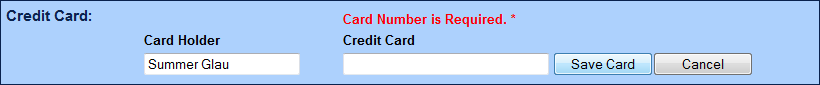


The list boxes also have the ability to remove a copy. To remove a copy, the user will just have to select a copy contained in the list box, which will then dynamically enable the Remove button. The user can then just simply press the remove button to remove a copy with the calculated fields being adjusted accordingly.The payment type consists of three options; cash, visa, and master card. If the cash option is selected the form will just totally ignore the credit card. If either one of the credit card payment types are selected, it will then show another section which will list the credit cards that correspond to the customer selected with the ability to select a card from the dropdown list (reflecting the change into another read only textbox called card holder.

If the card desired is not on the list, then the user has the ability to create a credit card from the transaction associated with the corresponding customer. The user just needs to press on the Add Card button, and it will then show a completely different section allowing the ability to insert a new credit card record.



The insertion of a new credit card can be done by just filling out the related fields. The card holder is automatically filled in corresponding to the customer object. If the card holder is different, then it can be changed. The credit card can only be a 16 digit number. It then can be saved by pressing on the Save Card button, which will then reshow the credit card selection dropdown list with the newly created card as the selected item. If the user presses the Cancel button, it will return to the previous section without making any changes.

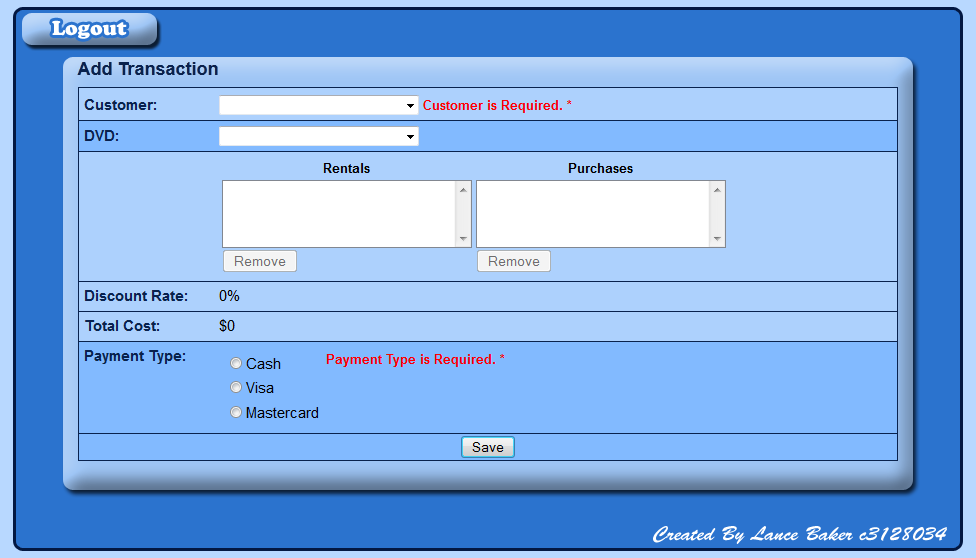


### Add Transaction

The add transaction page can be accessed from the menu under “Create New Transaction”. It will use the same form user control that is shown when displaying a record, except the transaction object is not displayed in the form. The save method accommodates this, and will instantiate a new object instead of retrieving it from the list. It will then add the new object to the list. Once the object is added, the page will refresh, which will allow further editing of the object.

The required field for creating a new transaction is that the customer must be selected, the transaction as DVD copies added to either the rentals or purchases, and that the payment type is selected (with also a credit card selected if the credit card type is chosen).

Since the DVD copies can either be added to the rentals & purchases (with either being valid if the other one empty) then the validation will be done on save instead, showing an error message indicating that it has failed if both are empty.



## DVDs

The DVDs page can be accessed from the menu under “View DVDs”. It lists the DVDs that are currently stored in the system. It uses a DataGridView which is bound onto the DVDs’ object list. It is comprised of functionality to view and delete a record. Viewing a record will redirect the user to another page to view a specific DVD object.

A DVD object consists of an internal structure containing the DVD copy information; which includes the barcode of the physical copy, and also a field indicating the availability of the copy. This allows the system to keep track on which DVD copies get loaned out, with also facilitating the ability to return an individual copy (by setting the availability back to true).

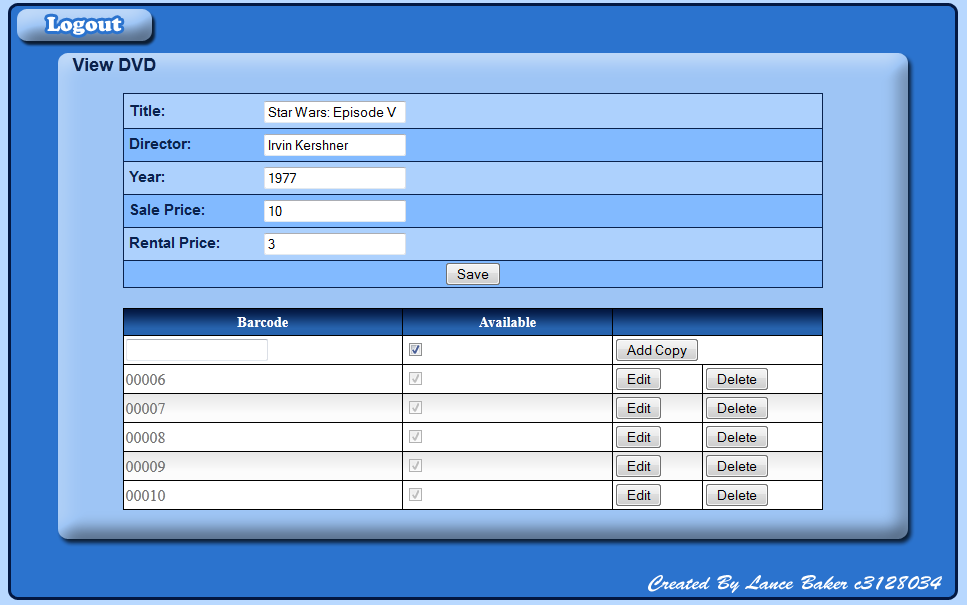


### View DVD

The View DVD page is accessed from the parent DataGridView in DVDs’ page by clicking on the View button. It is passed a unique identifier allowing the DVD to be retrieved from the internally stored list. Once it has the object, it is then passed into a method on the form user control to show the data on the form.

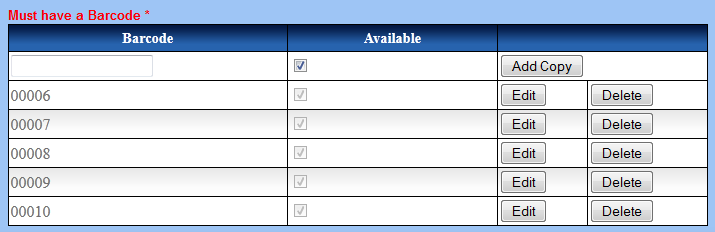
The form consists of a series of textboxes which are all required fields. The director field also consists of another check to enforce that it only consists of letters; the year has a validation rule to enforce that it must be four digits; and the price fields also have a rule stating that it must be numeric.

The DVD copies relating to the DVD object is also shown in a DataGridView; comprising of the functionality to add a new copy, edit an existing one, and also to delete one altogether.

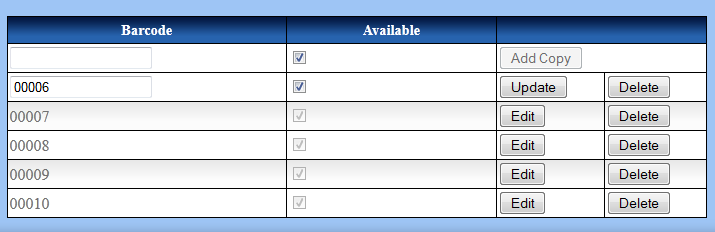


The DVD copy DataGridView lists the copies that the title currently has in the store. The copy details must have a barcode (which relates to the one physically placed onto the DVD) and also an indication whether it is currently available. If the copy is being rented out, or even purchased; then the availability field will be updated to false. Once a copy has been returned, then the copy can be updated to being available again, which will enable the ability for it to be rented-out or even purchased again.

The DataGridView comprises of the ability to Insert, Edit, and Delete records. The “Add Copy” button validates the barcode input field, and will display a message directly above the column relating to the rule. The barcode is required, and will only accept a number with five digits. The available field is not validated, since it can either be checked or unchecked. Once the “Add Copy” has been pressed, the new object will be instantiated and added to the internal list. The DataGridView is then refreshed to reflect the changes made.



A DVD Copy can be changed by pressing on the Edit button; which then will disable the “Add Copy” button until the Update button is pressed to make the changes to the object. If invalid data is entered; then an error message is shown on-top of the form once the Update button is pressed, it will stay in update mode allowing the user to then correct the changes. There isn’t any inline validation on the Edit record ability, since it cannot be applied to DataGridView fields. The validation rules for Editing a record is the same as Inserting; checking the barcode to ensure it is filled in and to ensure that it consists of a five digit number.



### Add DVD

The Add DVD page can be accessed from the menu under “Create New DVD”. It will use the same form user control that is shown when displaying a record, except the DVD object is not displayed in the form. The save method accommodates this, and will instantiate a new object instead of retrieving it from the list. It will then add the new object to the list. Once the object is added, the page will refresh, which will allow further editing of the object. It will then also show the DVD copies’ DataGridView, allowing the ability to insert copies on the newly created object.

The validation on the form is the same as outlined in View DVD. All fields are required, with the director having a restriction on only accepting alphabetic characters. The year will only accept a four digit number. The price fields can only accept numeric values (optionally allowing a decimal point).

