# Design Overview for The Record Store

Name: Dylan Armstrong

Student ID: 105040962

# Summary of Program

The record store application that I am building is an application in C# that should talk to a database to fetch a list of records available for purchase. The records will be data that should be able to be filtered, searched, add/removed to cart and should be able to have the quantity of that record updated, as well as a Boolean of whether the record is in stock or not.

The program should have a frontend, backend and database developed using the MVC (Model-View-Controller) architecture in C# with ASP.NET Core. The record application should have several different classes for things such as Records, the Cart, etc. The program should use polymorphism, inherited classes, and many more key OOP concepts.

The application will use scaffolded Db context items using the Microsoft Entity Framework package, for data access.

Include a sketch of sample output to illustrate your idea.

# Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Record Class (Model) | Int id -> Int  String artist -> String  String album -> String  Double price -> Double  Int quantity -> Int  Bool available -> Bool | The record class should contain properties for the things mentioned on the left, including different data types, for artist/album, price and if the record is available and in stock. |
| Cart Class (Model) | List<Record> items -> List<Record>  Double totalPrice -> Double | The cart class should contain the list of records in the cart, and the total price of all items in the cart. |
| RecordController Class (Controller) -> Inherits Controller Class in MVC | RecordStoreContext \_context -> RecordStoreContext  Task<IActionResult> Index() -> returns View with \_context.Records  Task<IActionResult> Find(string? artist, string? album) -> returns View with found \_context.Records  Task<IActionResult> Filter(enum FilterBy) -> returns View with filtered \_context.Records | The Record Controller is the controller for all things Records. It should be able to Get All Records via the Index method, Find Records and Filter Records. Filtering records is done based off an enum of what the user wants to filter based off, the find records has an optional artist OR username, one of these or both must be used (logic will be in the method itself to work this out). Index, gets every record on startup. |
| CartController Class (Controller) -> Inherits Controller Class in MVC | Task<IActionResult> Index -> returns View with \_context.Cart  Task<IActionResult> Add(int id) -> returns View \_context.Cart  Task<IActionResult> Remove(int id) -> returns View \_context.Cart  Task<IActionResult> UpdateQuantity(int id) -> returns View \_context.Cart | The Cart Controller is the central point for controlling routes related to the Cart. It has 4 methods, GetCart (which gets everything in the cart and creates a cart initiailly). Remove, removes an item based off id, Add, adds an item based off id and UpdateQuantity, updates the quantity of that item in the cart. |
| RecordStoreContext (DB Context) -> Inherits DbContext Class | DBSet<Record> Records -> DbSet<Records> | This class makes it so that Entity Framework can be used with the Record class and for a database access layer to be established with SQL Server. |
| DBInitalizer Class | Initialize(RecordStoreContext context) -> Add initial records to context | This class initializes the records into the database initially. |

Table 2: <<enumeration name>> details

|  |  |
| --- | --- |
| Value | Notes |
| Enum FilterBy -> price, artist, album, inStock | This enumerator is the different types of filters for the Filter method in the Records Controller. |

# Class Diagram

NOTE: RecordController and CartController inherit the built in Controller class. Also, the RecordStoreContext class inherits the DbContext class.

# A black screen with white rectangles Description automatically generated

# Sequence Diagram

Provide a sequence diagram showing how your proposed classes will interact to achieve a specific piece of functionality in your program.