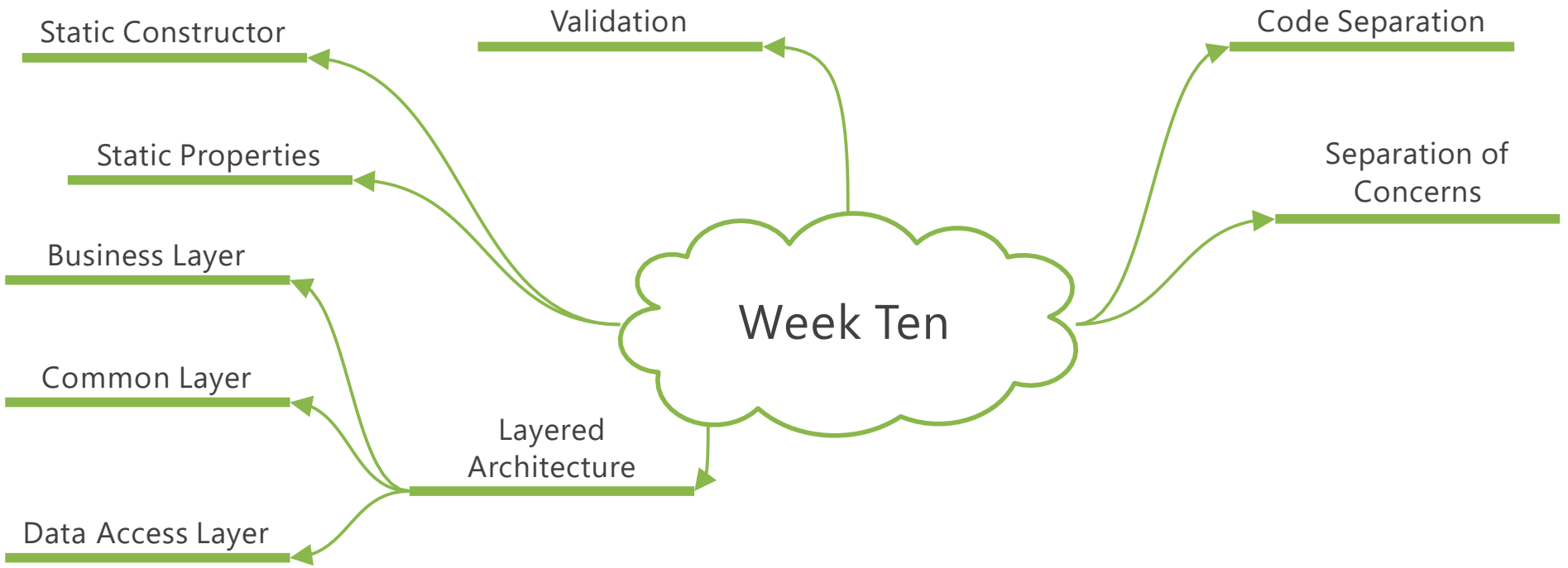
COMP 2614

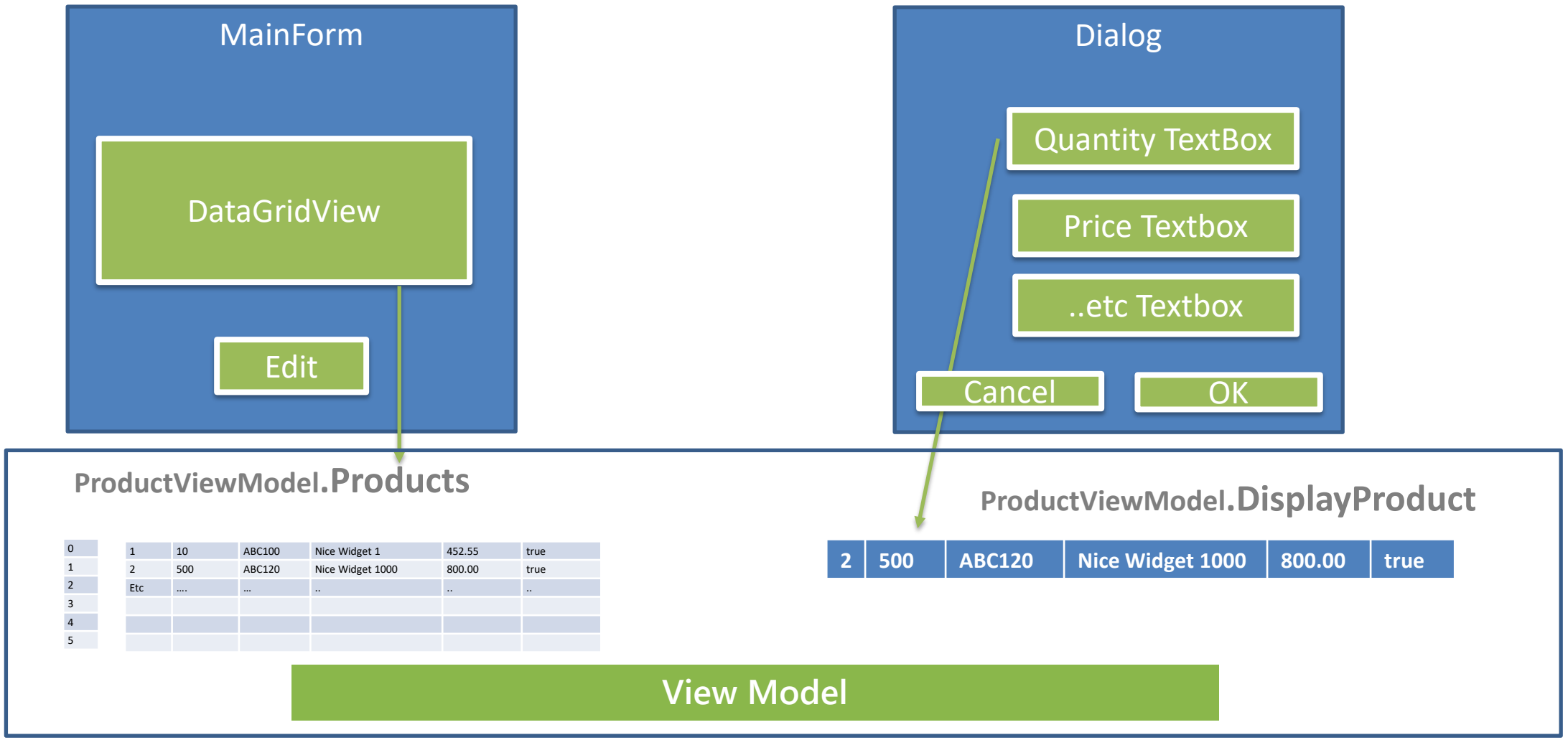
C# Application Development

Week Ten

Tonight's Learning Outcomes



Product ViewModel – Lab 9



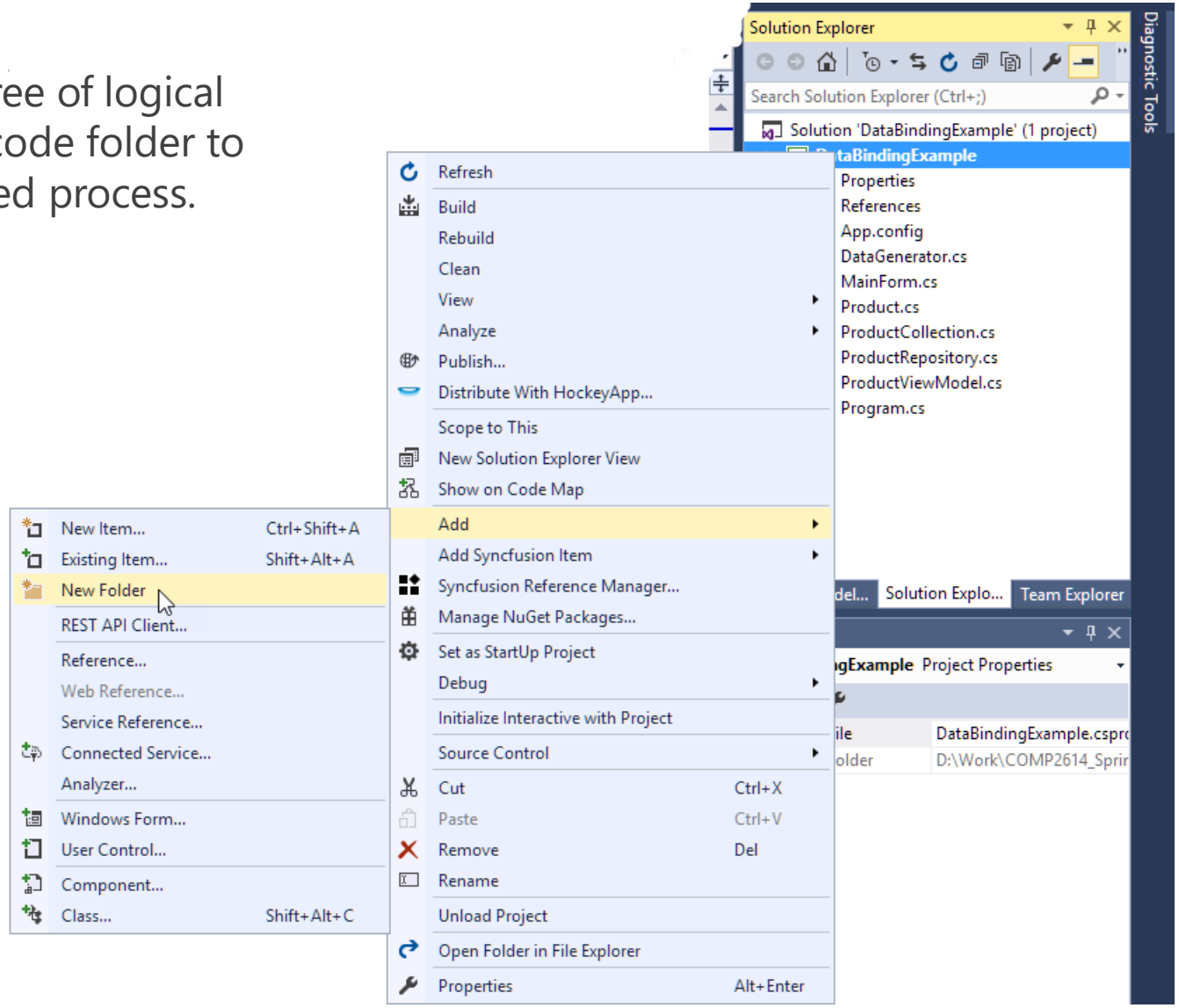
Product ViewModel – Lab 9

Sequence of events:

- Row is selected, index is assigned to current row index
- Edit Button is clicked
- viewModel.DisplayProduct is set w/ productVM.products[index]
- Dialog is instantiated
- Dialog.ViewModel is passed a reference to the view model
- setBindings() in Dialog for controls on dialog
- OK Button is clicked
- MainForm handles Dalog.DialogResult == DialogResult.OK
- productVM.Products[index] is updated to be DisplayProduct
- Or, new product is added to the list

Separation of Concerns – Code Separation

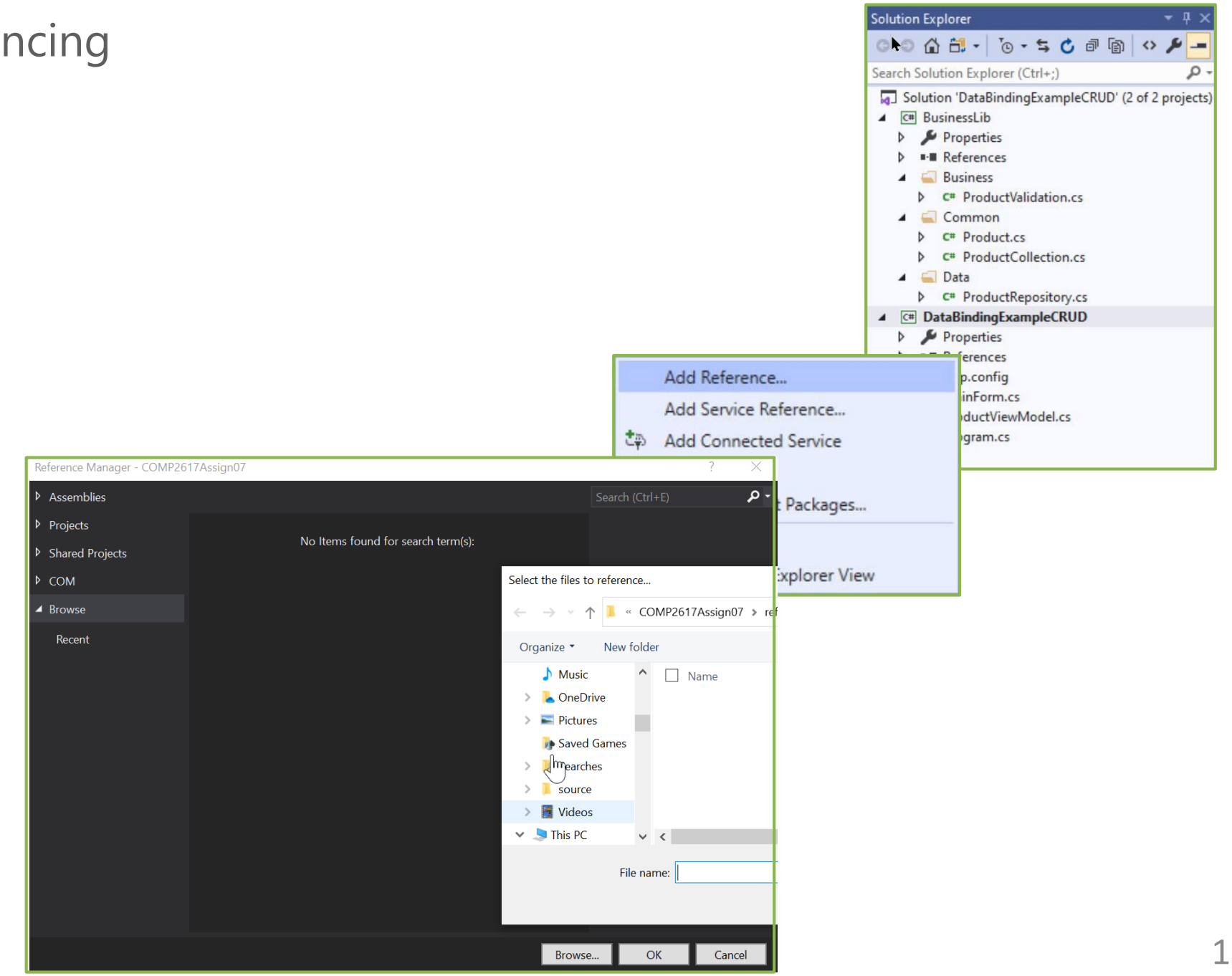
We can implement a higher degree of logical separation by creating a source code folder to separate the classes of a dedicated process.



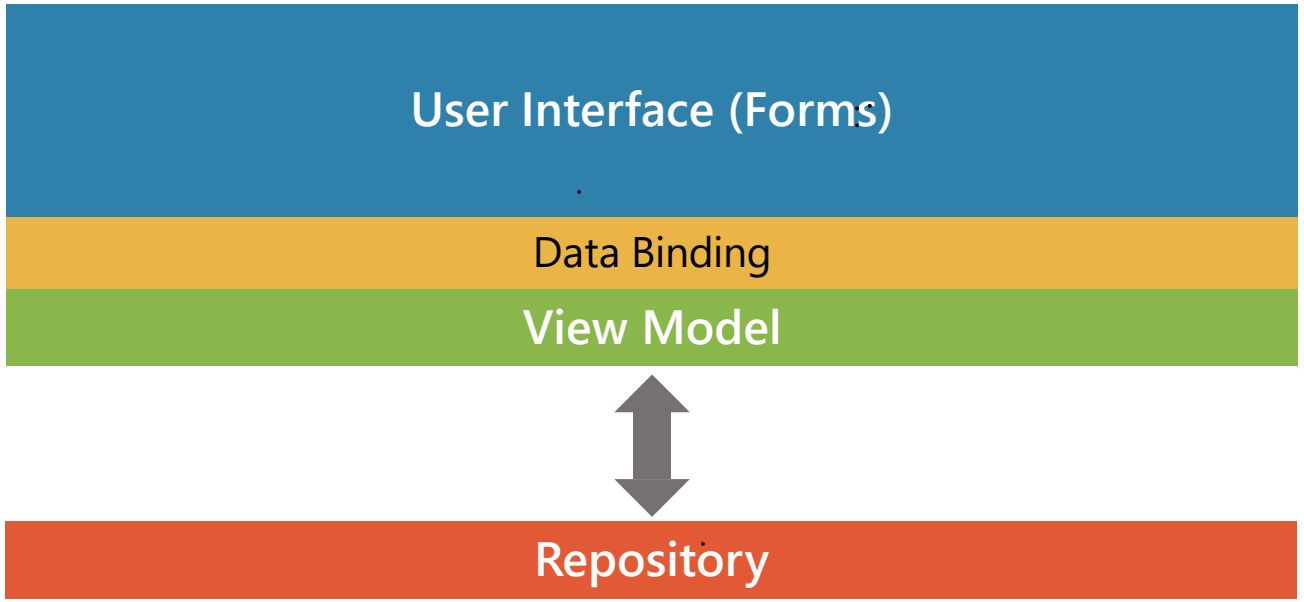
Class Libraries - Referencing

Referencing

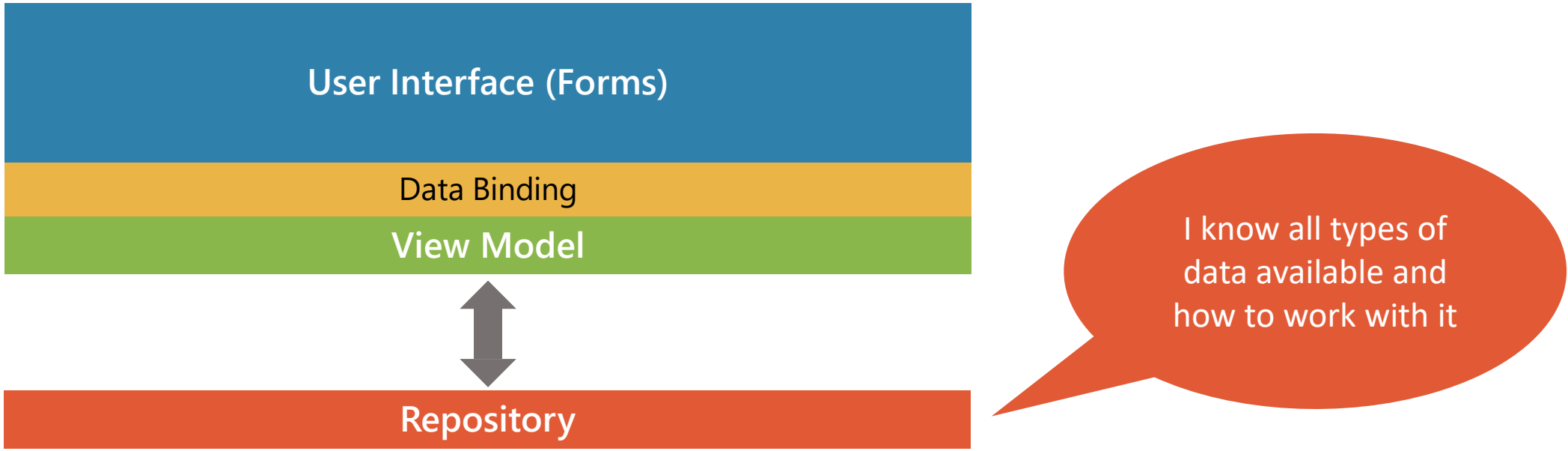
- Right-click on References in the UI Project in the Solution Explorer
- Click Add Reference…
- Click Browse
- Select the dll to add to the Project (In this case, just the one)
- Click OK



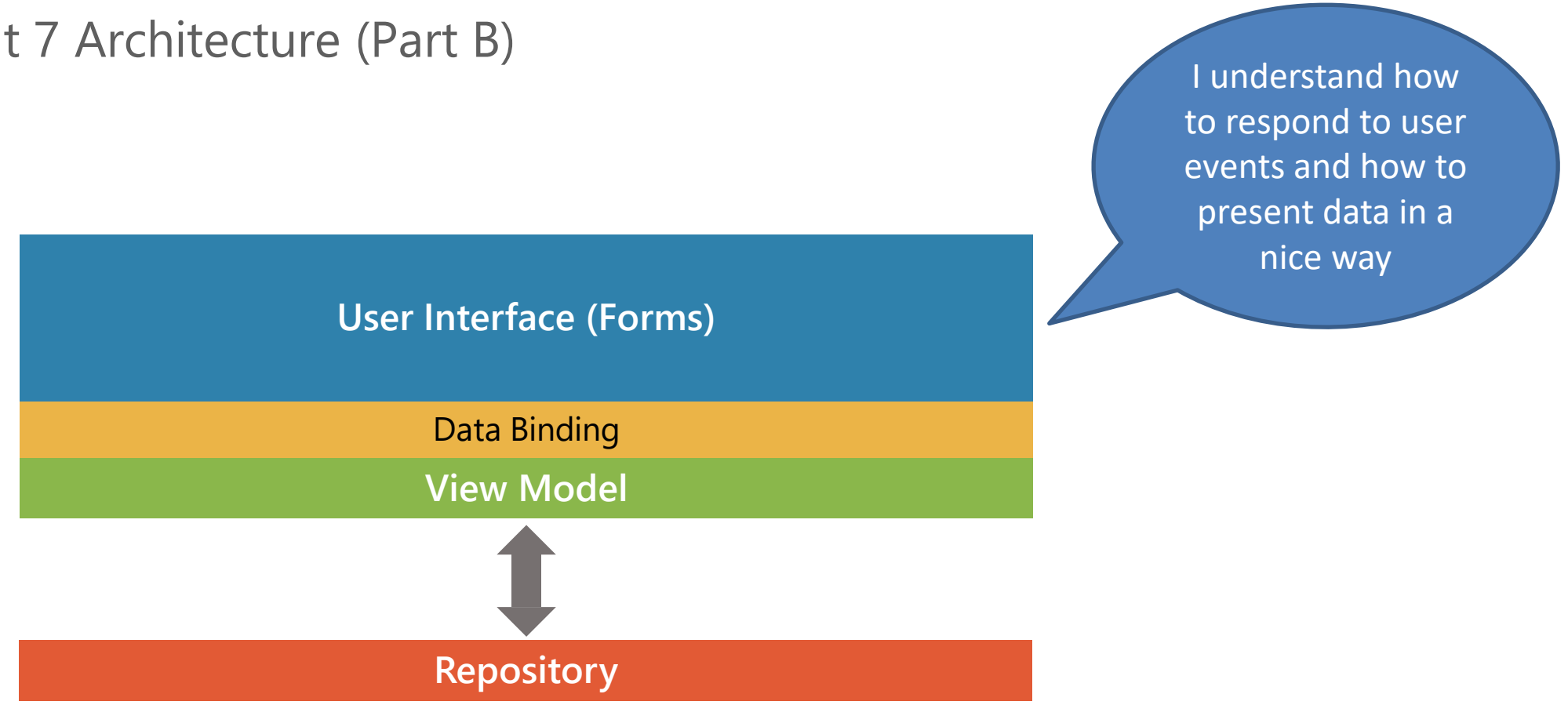
Assignment 7 Architecture (Part B)



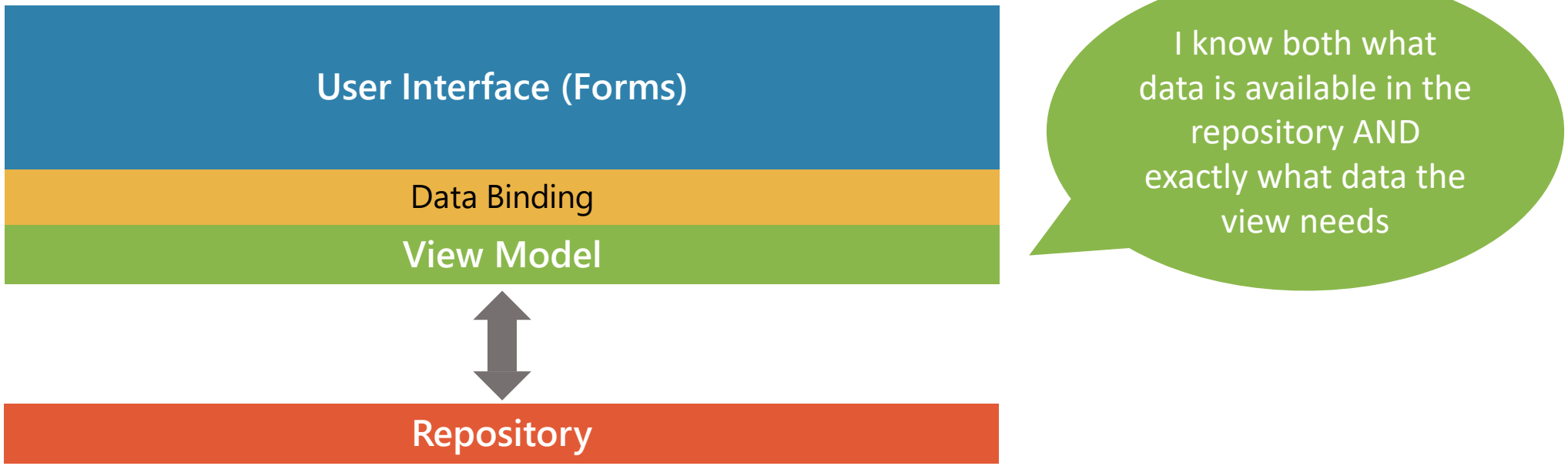
Assignment 7 Architecture (Part B)



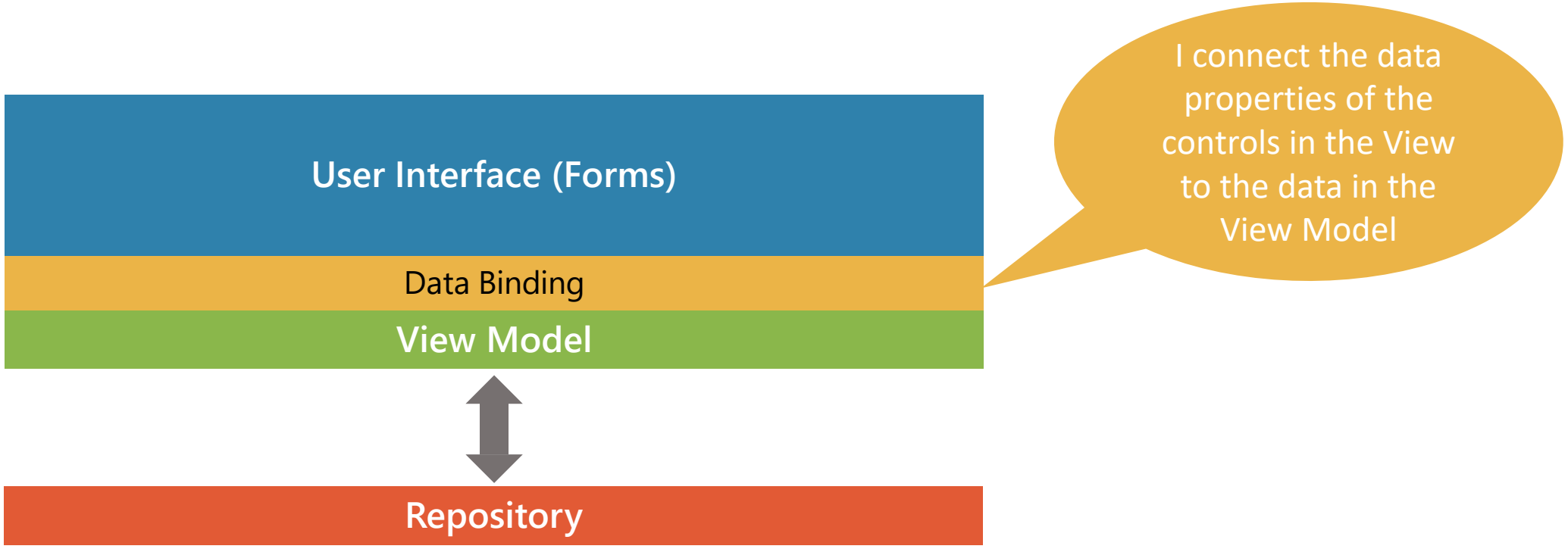
Assignment 7 Architecture (Part B)



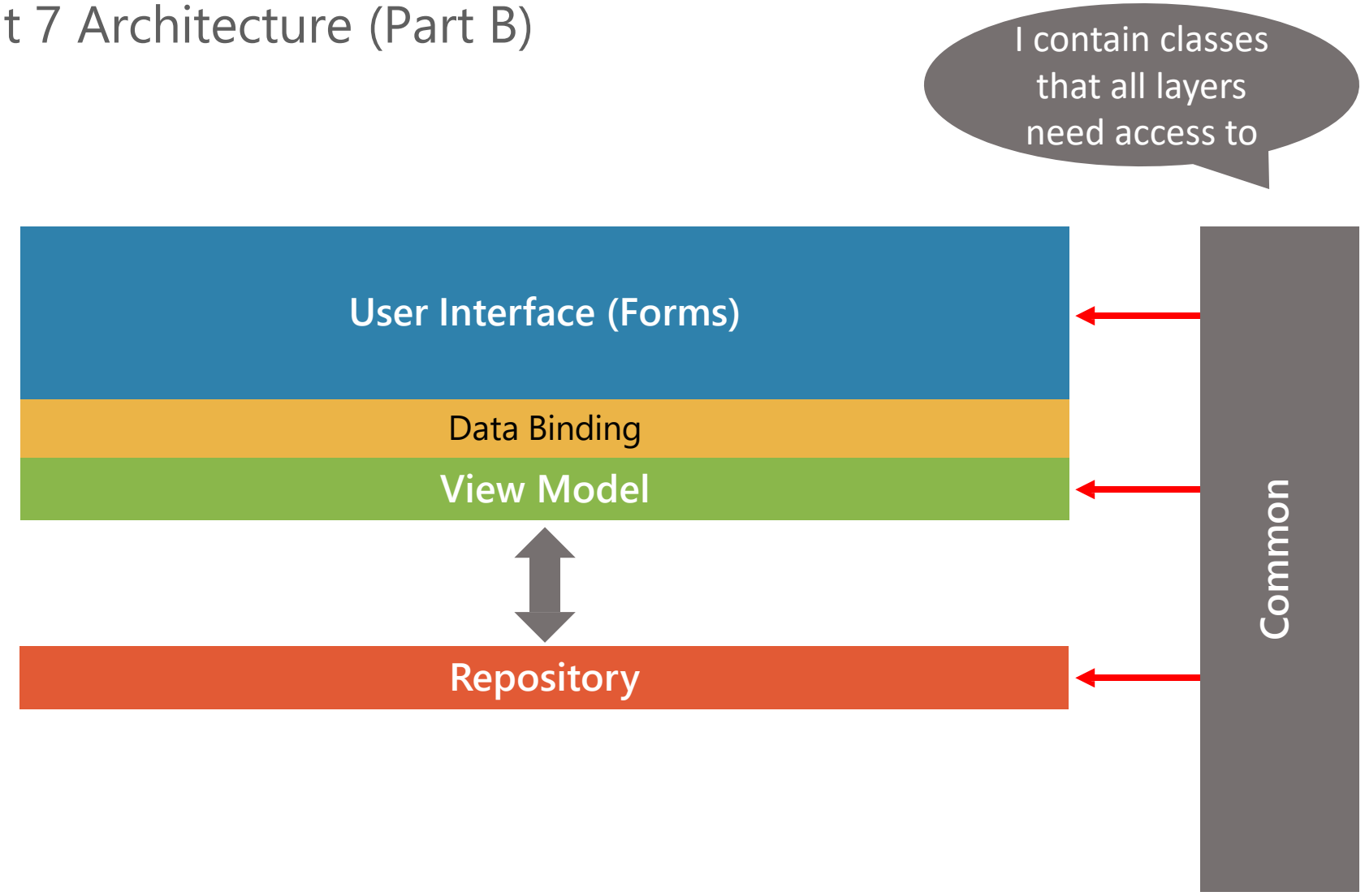
Assignment 7 Architecture (Part B)



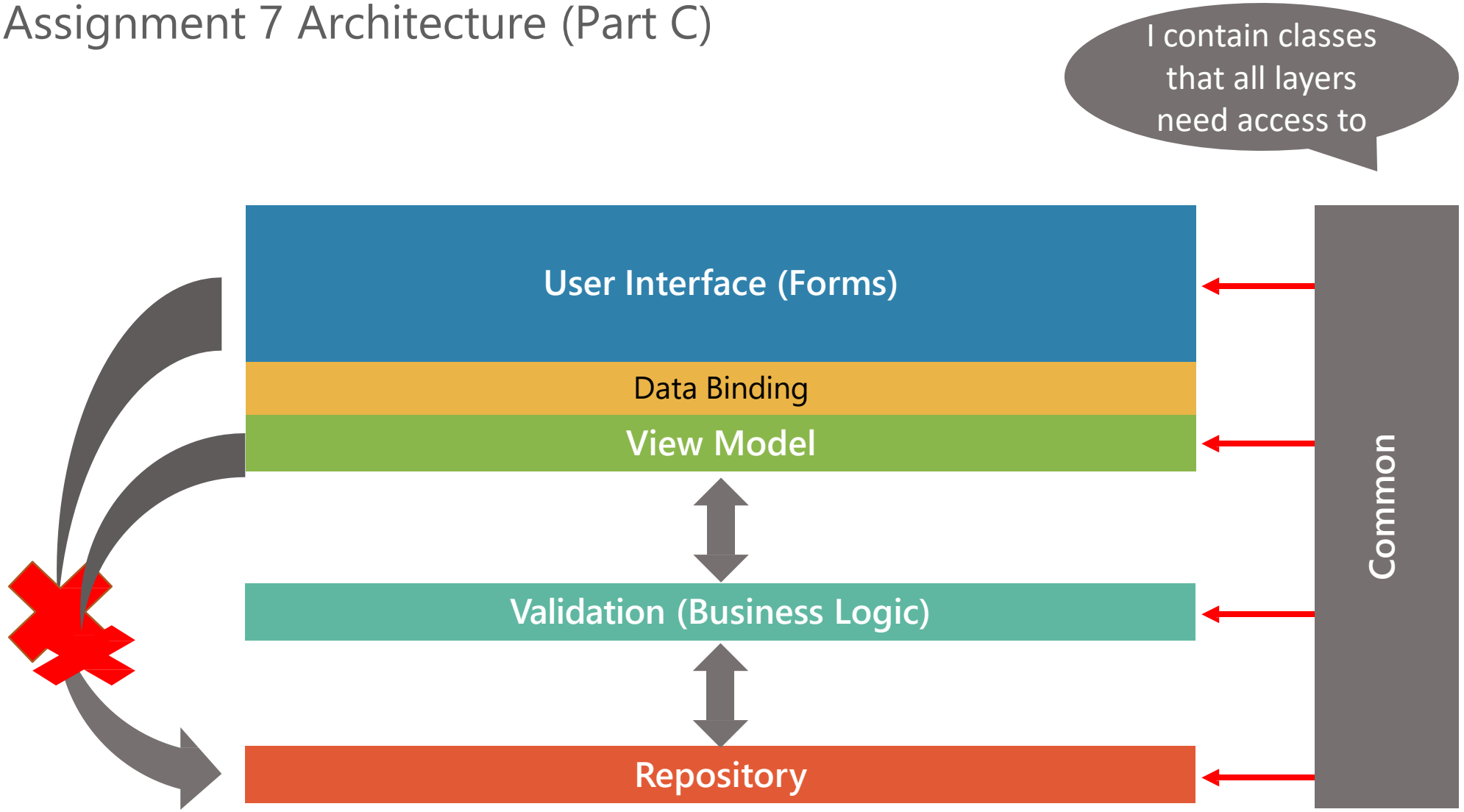
Assignment 7 Architecture (Part B)



Assignment 7 Architecture (Part B)



Assignment 7 Architecture (Part C)



Validation Layer

```
1 reference
public static int AddProduct(Product product)
{
    if (validate(product))
    {
        return ProductRepository.AddProduct(product);
    }
    else
    {
        return -1;
    }
}
```

- The Validation Layer has Add/Update methods like the repository. The Validation methods are inserted in between the UI and Repository classes.
- A validation method is called to test the business rules for the data object.
- The Validation Layer forwards the RowsAffected from the Repository to the UI layer via a return.
- The Repository method is not invoked when a validation failure occurs.

Validation Layer

Business Rules are defined and enforced here

```
2 references
74      private static bool validate(Product product)
75      {
76          bool success = true;
77          errors.Clear();
78
79          if (product.Quantity < 0)
80          {
81              errors.Add("Quantity cannot be less than zero");
82              success = false;
83          }
84
85          if (product.Cost < 0.00m)
86          {
87              errors.Add("Cost cannot be less than zero");
88              success = false;
89          }
90
91          if(product.SellPrice < product.Cost)
92          {
93              errors.Add("SellPrice cannot be less than Cost");
94              success = false;
95          }
96
97          return success;
98      }
99
```

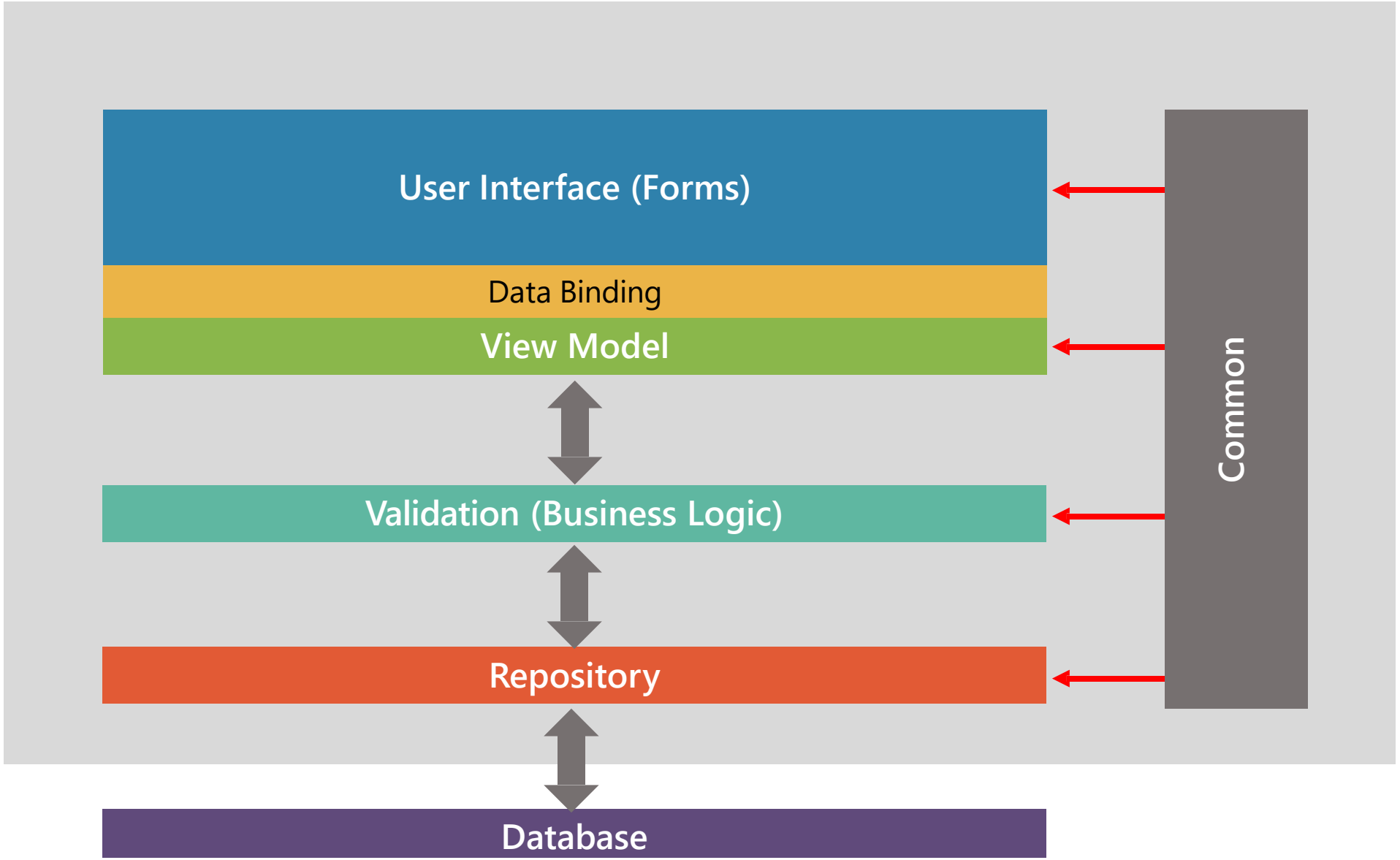
Static List<string> declaration

```
6 references
11      class ProductValidation
12      {
13          private static readonly List<string> errors = new List<string>();
14
15
```

Static Property to expose ErrorList

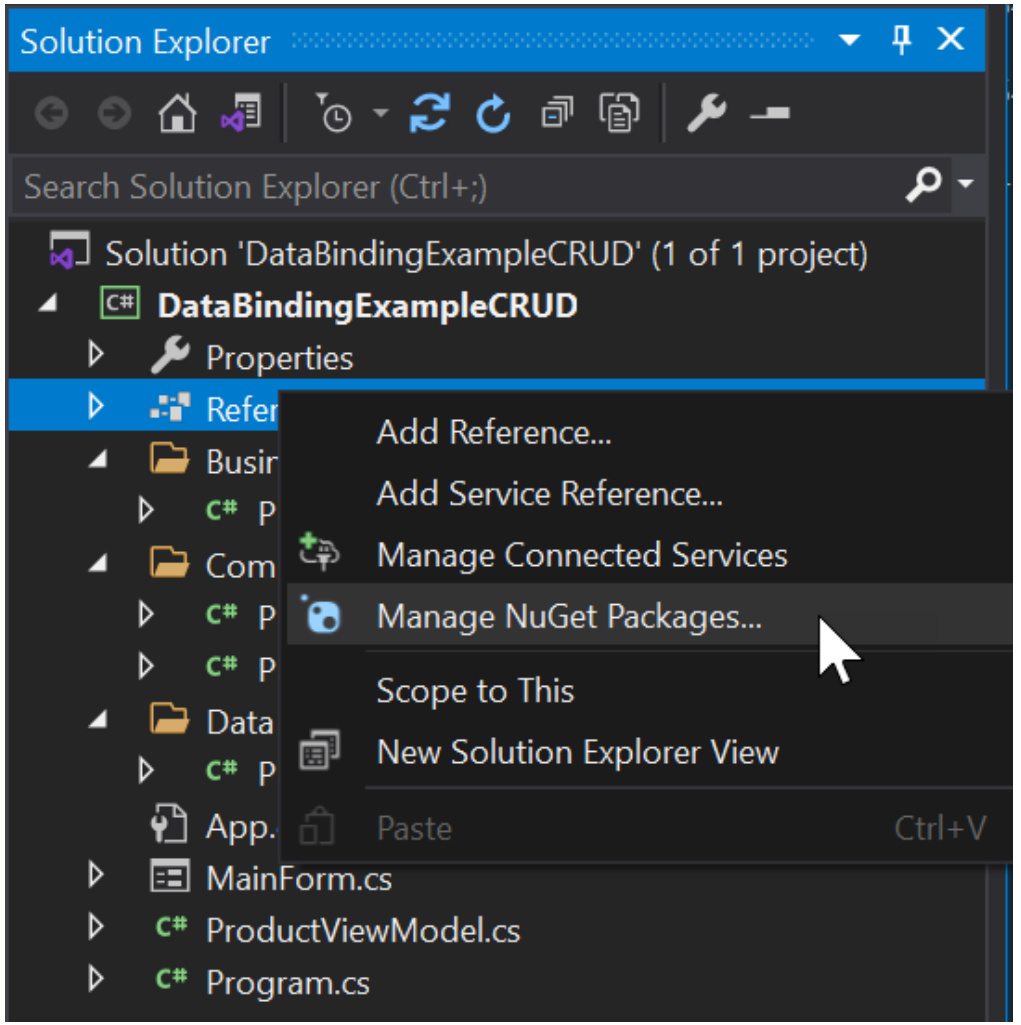
```
1 reference
30      public static string ErrorMessage
31      {
32          get
33          {
34              string message = "";
35
36              foreach (string line in errors)
37              {
38                  message += line + "\r\n";
39              }
40
41              return message;
42          }
43      }
44
```

Assignment 7 Architecture (Part C)



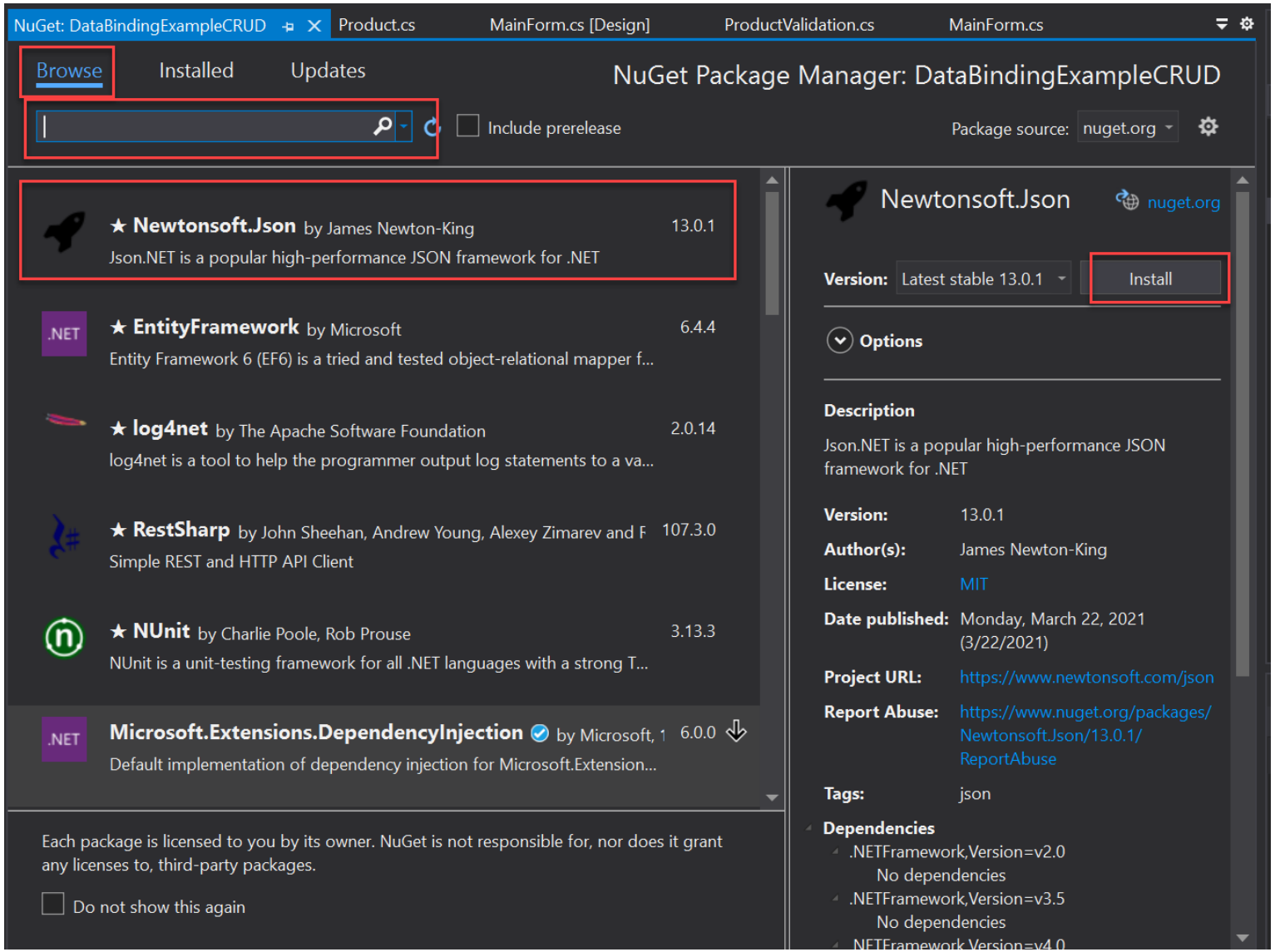
NuGet Packages

- NuGet packages are zip files containing any files required for a particular feature to work
- The packages are then published and can be added to any project through the Nuget Package Manager
- To access the NuGet package manager, right click “References” in solution explorer, then select “Manage NuGet Packages”
- From there, you can search for and install packages



NuGet Packages

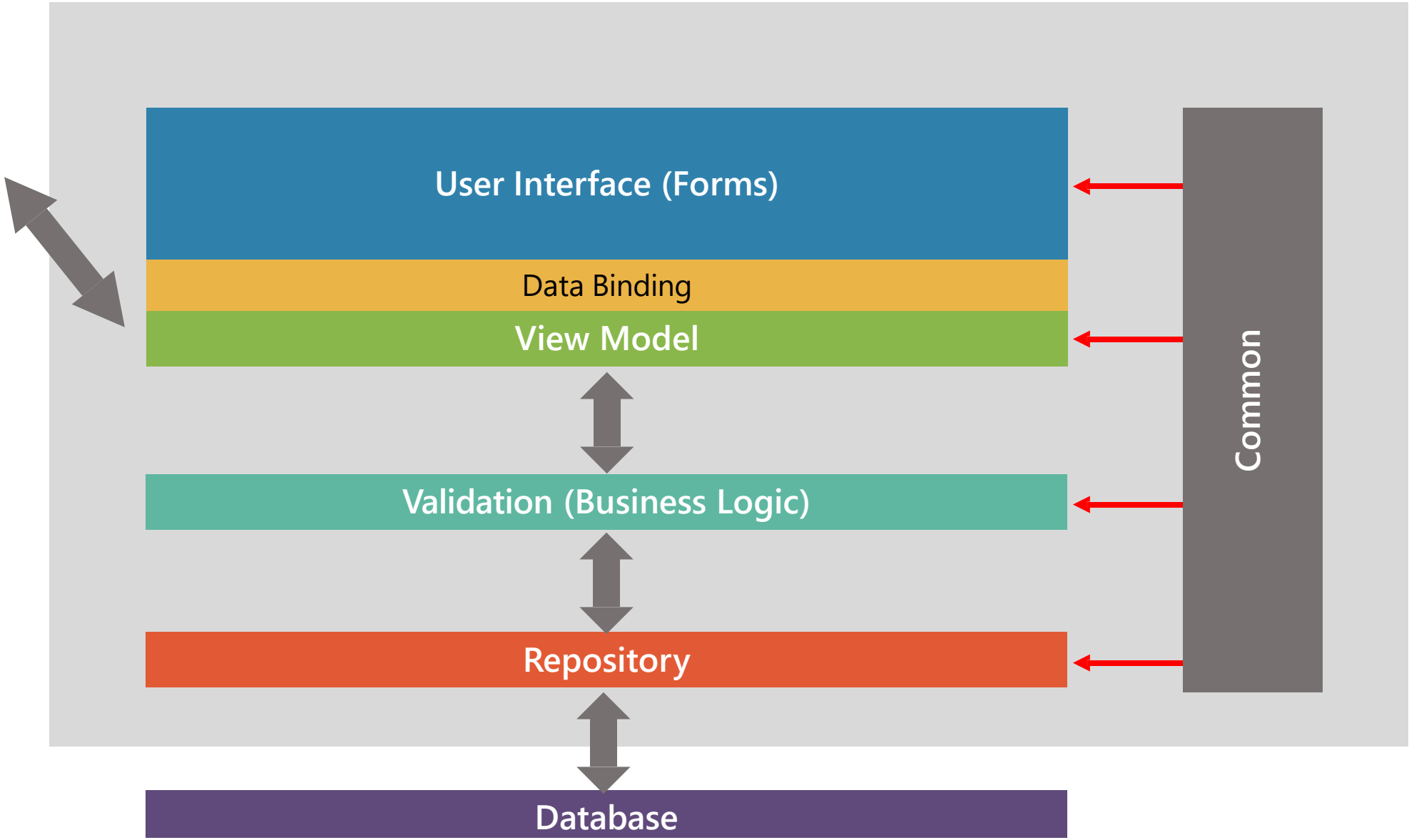
- In the package manager, you can search for and install the package you need for your app
- There is also a command line tool available, if you prefer that to the GUI package manager



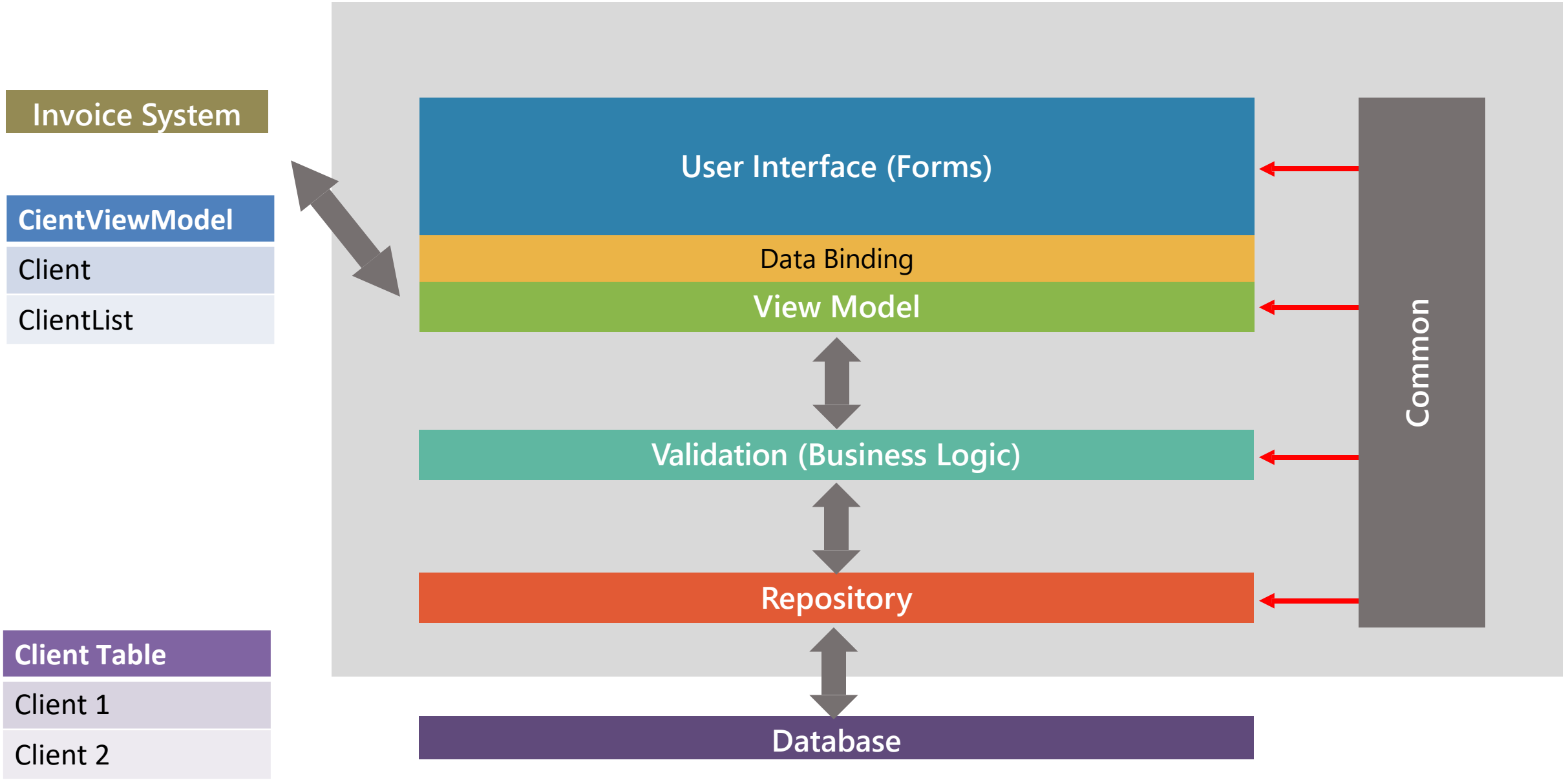
Assignment 7 Architecture (Part C)

Invoice System

The Invoice System is a web service that returns a list of Invoices when given a client code.



Assignment 7 Architecture (Part C)



Assignment 7 Architecture (Part C)