.NET MVC apps have Automatic tools added into the wwwroot folder alongside the Boostrap CSS libraries that are automatically added to the project.

JQuery validate adds validation to our project
JQuery Validate unobtrusive allows for client-side validation

It is best to use both when able but server side validation is the more important of the 2 – can't be overridden by user in browser.

Adding using System.ComponentModel.DataAnnotations; allows us to add annotations to our model properties to allow for automatic validation of data when entered by our forms.

The attributes will manage the user input on the client side and throw errors if the validation rules are not met.
Whenever the Model.IsValid() method is called in our controllers it checks our data against the annotations in the server if the client side validation is somehow bypassed. If the data is not valid it will not be processed.

See examples:
[Display(Name = "Joke Answer")]
Allows us to specify the column/attribute name displayed in our browser

[Required]
Makes the field mandatory to be filled in. This one may not be needed if using Entity Framework.

[StringLength(20, ErrorMessage = "Your answer can't be that long!")]
Sets the max length of a field and the message if caught by server-side validation. Error will not appear if handles by client side.

[StringLength(maximimLength: 2 MinimumLength = 20)]
Sets the both the minimum and max length of a field. Note the way the max length is added using a colon (:) instead of an equals sign(=). This is because it is passed to the constructor of the underlying class of the annotation. An error message could also be added the same as the example above this one.

[Range(10, 20)]
Sets the min and max range allowed to be entered into the field. This only works on numeric data types

[DataType(DataType.{type})]
There are several other datatype options that can be used in the previous example, just specify the type after the dot in the annotation. The following options are examples that can be used here:

- **DateTime-** Forces the entry to be data/time format, Also changes input field in view to date/time picker
- **Date** – Same as above but only gives a date picker.
- **Password** – Makes the field a password field which only shows dots instead of the text characters
- **Email** – Makes sure the content of the field is a valid email (using simple email rules)
- **PhoneNumber** – Ensures the field holds a valid phone number

NOTE: Some of these might not work in all applications depending on .NET version and setup.

[RegularExpression(@"^[a-zA-Z''-'\s]{1,40}$")]
Allows you to insert a regular expression to check if the data meets the required rules. This can often be used as an alternative to the phone and email datatypes if more specific rules are needed or they are not working using the Datatype annotation shown above.
NOTE: You need to be able to write/source regex for this annotation.

USING VIEWBAG
A state management object used to work around the statelessness of the Web App that helps pass data from the controller to the view. Can only be inserted from controller and read in the view. Unlike session data which can reads/write at both sides. Only valid during current HTTP request.

```
ViewBag.ErrorMessage = "The data entered was invalid!";
```

Enter above after IsValid check to enter message to ViewBag. ErrorMessage name shown is property to be added to ViewBag. Use code below to add heading to html to display if message is not empty.
NOTE: Test on valid entry to demonstrate.

```
@{
   string message = ViewBag.ErrorMessage;
}

@{
   if (String.IsNullOrEmpty(message) == false)
   {
      <h2 class="text-danger">@message</h2>
   }
}
```