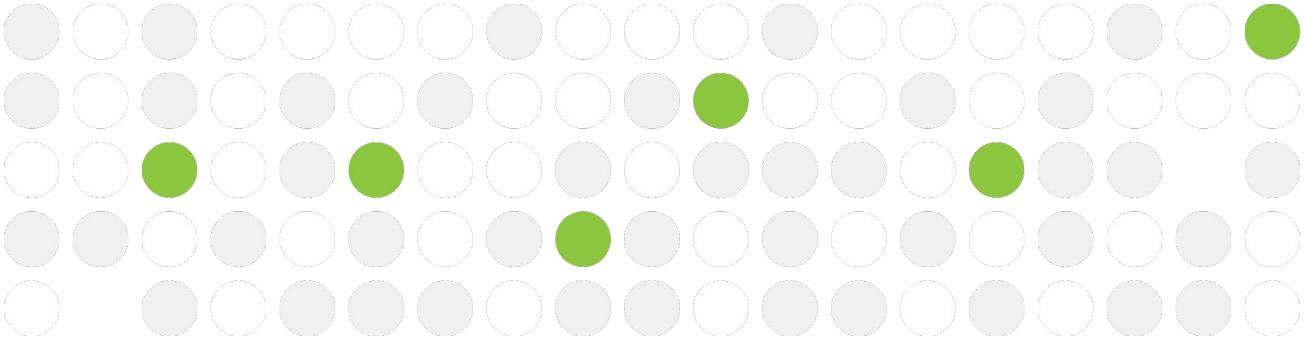
EXPERT NETWORK

Web Development with ASP.NET Core 7



S3

FII Practic Project Presentation (1st of April)

- In our last session you will be invited to present your own project for this course.
- You may choose any subject for your project, other than cars, and it needs to be your own creation.
- The best 3 projects will receive Prizes and points on the Internship Technical Test.

EXPERT NETWORK

Internship Points

To qualify for the advantage in the Internship selection you will need to:

1. **Send your CV at hr@expertnetwork.ro**
2. **Register and Take the Technical Test (grades 0 to 10)**
3. **1st place will receive +5 points**

   2nd will receive +4 points

   3rd will receive +3 points.

EXPERT NETWORK

Internship

Our **Full Stack .NET Internship** will start soon.

Wanna join us?

- Send your CV at hr@expertnetwork.ro.
- Application Deadline: **March 19th**

For more details access our website: expertnetwork.ro

EXPERT NETWORK

The Sessions

1. Data Access
2. Concepts and Techniques
3. ASP.NET Core Introduction
4. ASP.NET Core Advanced
5. Deploy in the Cloud

*Note that each session builds upon the previous one.

EXPERT NETWORK

For this session

- Finishing CRUD
- HTTP
- MVC Pattern
- What is ASP.NET Core
- Create Web Application

*We will start from https://github.com/AlexandruCristianStan/FII-Practic-2023.

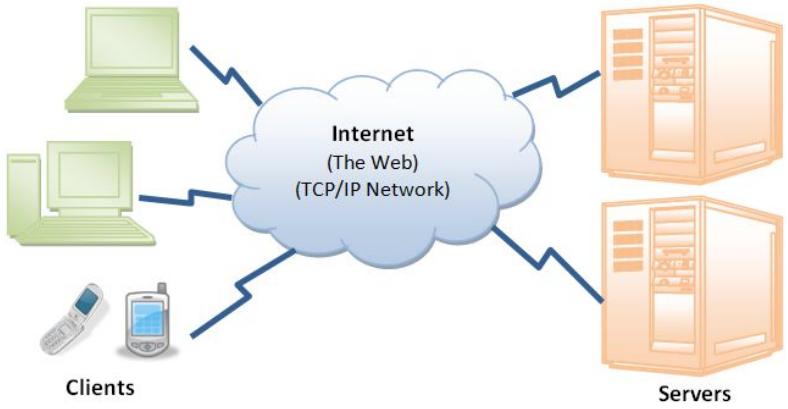
EXPERT NETWORK

Exercise 1: Finish the CRUD

The WEB

Internet (or The Web) is a massive distributed client/server information system

Many applications are running concurrently over the Web, such as web browsing, e-mail, file transfer, audio & video streaming, and so on. In order for proper communication to take place between the client and the server, these applications must agree on a specific application-level protocol such as HTTP, FTP, SMTP, POP, and etc.



EXPERT NETWORK

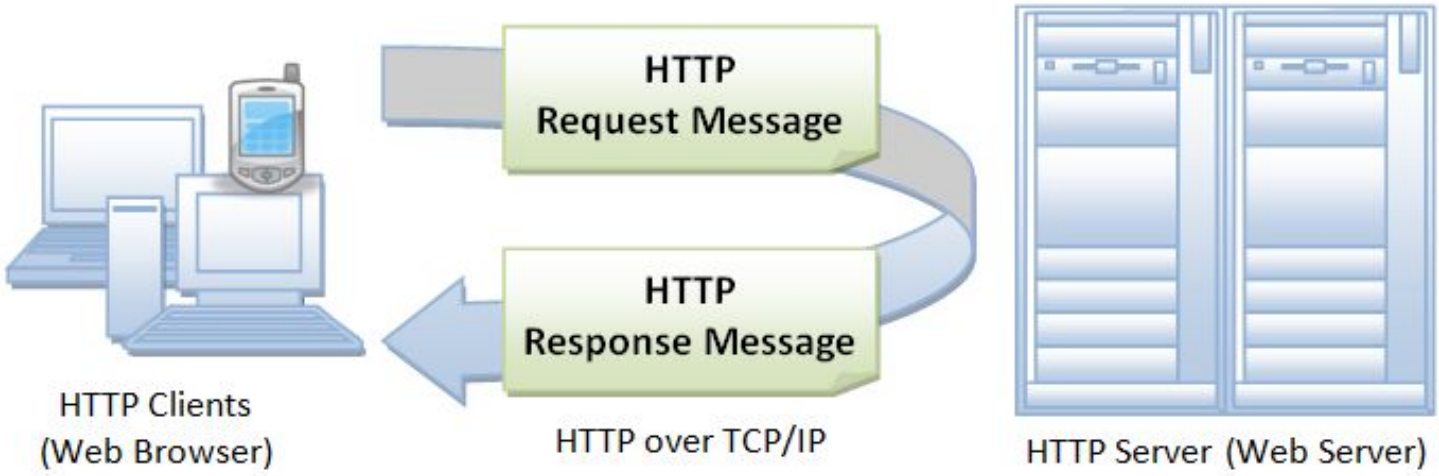
HTTP

HTTP is perhaps the most popular application protocol used in the Internet.

HTTP is an **request-response client-server** protocol.

An **HTTP client** sends a request message to an **HTTP server**. The server, in turn, returns a response message. In other words, HTTP is a pull protocol, the client pulls information from the server (instead of server pushes information down to the client).

HyperText Transfer Protocol



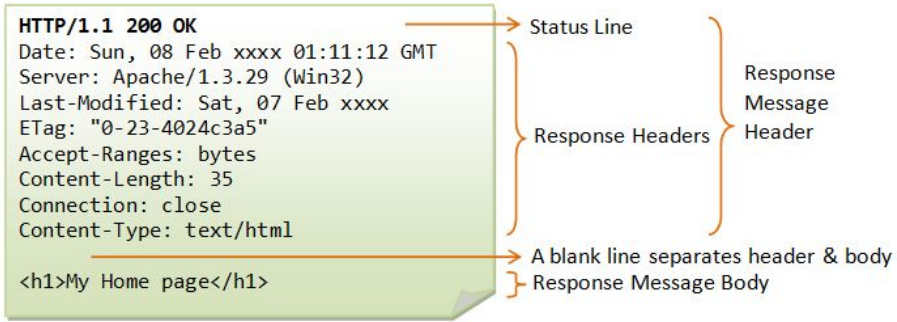
EXPERT NETWORK

HTTP

- HTTP is a **stateless** protocol. In other words, the current request does not know what has been done in the previous requests.
- HTTP permits negotiating of data type and representation, so as to allow systems to be built independently of the data being transferred (using headers).

HTTP Request / Response

Request



Response

EXPERT NETWORK

HTTP Request Methods

Methods	Meaning
GET	A client can use the GET request to get a web resource from the server.
POST	Used to post data up to the web server (Create)
PUT	Ask the server to modify the data (Full update)
PATCH	Ask the server to modify the data (Partial update)
DELETE	Ask the server to delete the data

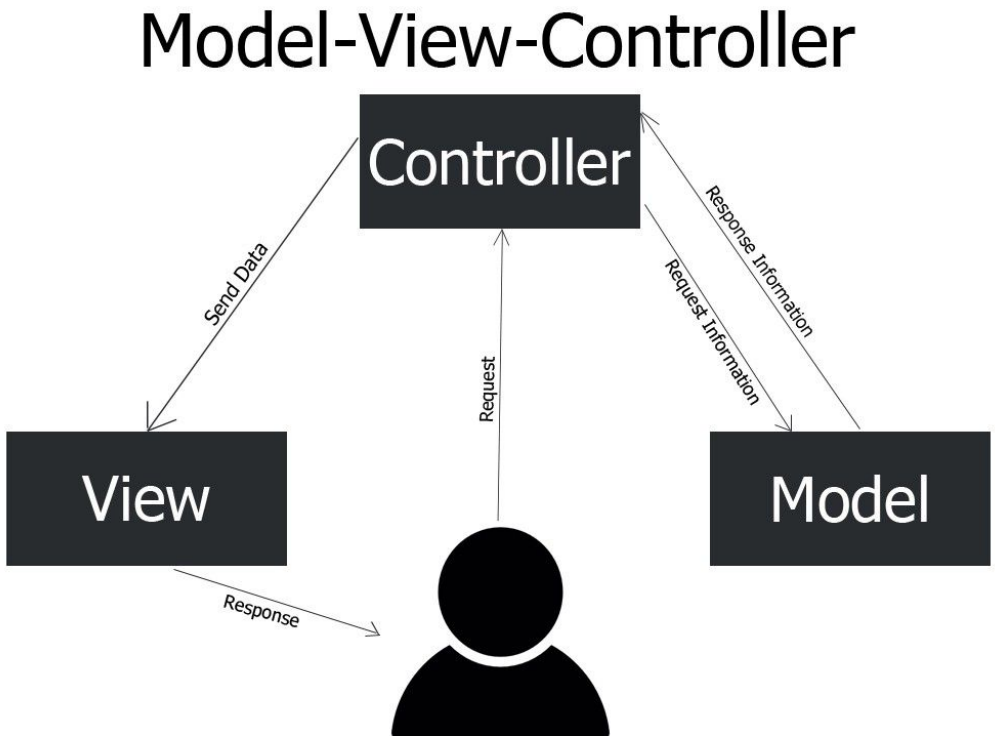
Exercise 2: Postman Echo

Model-View-Controller (MVC)

- Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

- Traditionally used for desktop graphical user interfaces (GUIs), this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate implementation of the pattern.

Model-View-Controller (MVC)



EXPERT NETWORK

C# vs .NET

- C# is a programming language
- .NET is an application framework runtime
- .NET allows you to run applications, allocate / deallocate memory, security, portability etc.
- Both are going hand in hand but you are not forced to stick with C#. There are other alternatives such as VB.NET (bleah) and F#

```
class Example { }
```

```
class Example
{
    static void Main()
    {
        // Here we call into the .NET to
        // write to the output console
        System.Console.Write("hello, world");
    }
}
```

EXPERT NETWORK

.NET vs ASP.NET (Core)

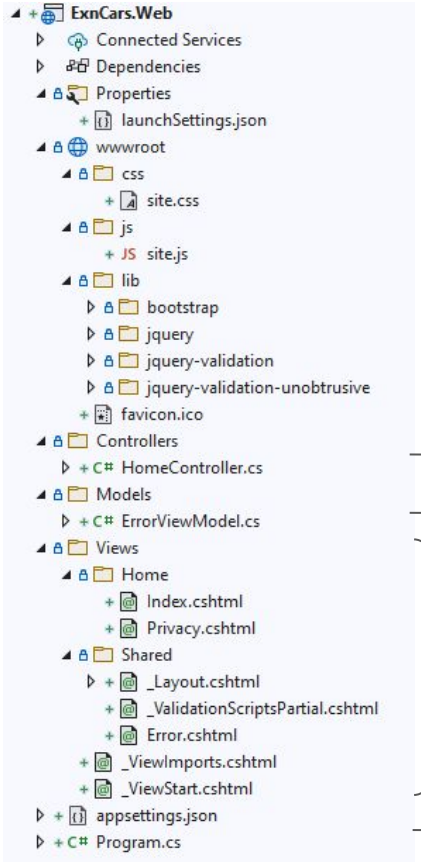
- ASP.NET Core is a high performance, open-source framework for building web applications
- It provides an easier way for you to create and maintain web applications.
- Microsoft took .NET APIs and created stuff for your in order to avoid to reinventing the wheel

Other frameworks

- Windows Forms
- WPF (Windows Presentation Foundation)
- WCF (Windows Communication Foundation)
- ASP.NET Web Forms
- ASP.NET MVC
- Universal Windows Platform
- Xamarin
- **.NET MAUI (Multi-platform App UI)**
- **ASP.NET Core (MVC / Blazor / Razor Pages)**

Exercise 3: Creating the web project

ASP.NET Core 7 MVC



Static files

Controllers

ViewModels

Views

Startup & Configuration

EXPERT NETWORK

Routing

- ASP.NET Core MVC uses **Routing** to map URLs of incoming requests to specific Controller Methods (also known as *Actions)*

```
app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

EXPERT NETWORK

Routing

The route template `"{controller=Home}/{action=Index}/{id?}"` matches a URL path like *localhost:port/Products/Details/5*.

Extracts the route values `{ controller = Products, action = Details, id = 5 }` by tokenizing the path. The extraction of the route values results in a match if the app has a controller named ***ProductsController*** and a ***Details*** action:

```
public class ProductsController : Controller
{
    public IActionResult Details(int id)
    {
        return View();
    }
}
```

EXPERT NETWORK

Routing

localhost:port/Books/GetBooksByAuthor?authorName=Oliver


```
public ActionResult GetBooksByAuthor(string authorName) {...}
```

EXPERT NETWORK

Views

In the MVC pattern, the *view* handles the app's data presentation and user interaction. A view is an HTML template with embedded **Razor** markup.
In ASP.NET Core MVC, views are *.cshtml* files that uses C# in Razor markup.

Views

```
@{
   var quote = "The future depends on what you do today";
}
<p>@quote</p>

@{
    var quote = "Hate cannot drive out hate, only love can do that.";
}
<p>@quote</p>
```

Renders:

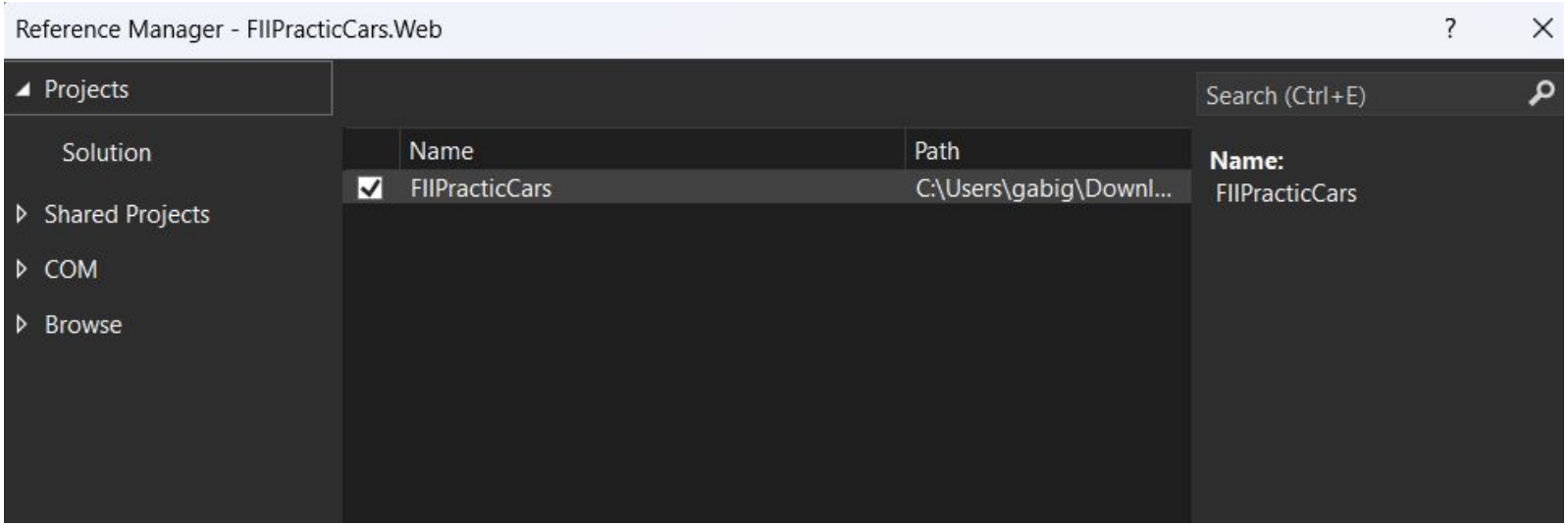
```
<p>The future depends on what you do today.</p>
<p>Hate cannot drive out hate, only love can do that.</p>
```

Razor

- Looping @for, @foreach, @while, and @do while
- Conditionals @if, else if, else, and @switch
- @try, catch, finally
- @using directive
- @model LoginViewModel (view model passed to a view)

Connect the projects together

Add a dependency for FIIPracticCars.Web to FIIPracticCars



EXPERT NETWORK

Connect the projects together

3. Modify FIIPracticCars.Web/Program.cs like:

```
using FIIPracticCars.Repositories;
using FIIPracticCars;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddDbContext<CarsContext>(options =>
    options.UseSqlServer("Server=GORGAN-PC02\\SQLEXPRESS;Database=FIIPracticCars;Trusted_Connection=True;Encrypt=True;TrustServe
    .LogTo(Console.WriteLine, minimumLevel: LogLevel.Information))
  .AddScoped<ICarsUnitOfWork, CarsUnitOfWork>()
  .AddScoped<IUserRepository, UserRepository>();

builder.Services.AddControllersWithViews();

var app = builder.Build();
```

EXPERT NETWORK

Exercise 4: Connect the projects together

Pass data to views

- Strongly typed data
  - ViewModel
- Weakly typed data
  - ViewData
  - ViewBag

Strongly typed data

- Using a ViewModel to pass data to a view allows the view to take advantage of *strong* type checking
- *Strong typing* means that every variable and constant has an explicitly defined type (*string, int, DateTime etc.)*
- The validity of types used in a view is checked at ***compile time***
- In VS & VS Code, it unlocks Intellisense

Strongly typed data

```
public IActionResult Contact()
{
    ViewData["Message"] = "Your contact page.";

    var viewModel = new Address()
    {
        Name = "Microsoft",
        Street = "One Microsoft Way",
        City = "Redmond",
        State = "WA",
        PostalCode = "98052-6399"
    };

    return View(viewModel);
}
```

```
@model WebApplication1.ViewModels.Address

<h2>Contact</h2>
<address>
    @Model.Street<br>
    @Model.City, @Model.State @Model.PostalCode<br>
    <abbr title="Phone">P:</abbr> 425.555.0100
</address>
```

EXPERT NETWORK

Weakly typed data

- Weak types means that you don’t have explicitly declare the type of the data you are using
- Used primarily when you want to pass small amounts of data in and out of controllers and views
- ViewData (dictionary) & ViewBag (dynamic) can be used for weakly typed data. Both objects are accessible in Controllers & Views

Weakly typed data

```
public IActionResult SomeAction()
{
    ViewBag.Greeting = "Hello";
    ViewBag.Address  = new Address()
    {
        Name = "Steve",
        Street = "123 Main St",
        City = "Hudson",
        State = "OH",
        PostalCode = "44236"
    };

    return View();
}
```

```
@ViewBag.Greeting World!

<address>
    @ViewBag.Address.Name<br>
    @ViewBag.Address.Street<br>
    @ViewBag.Address.City, @ViewBag.Address.State @ViewBag.Address.PostalCode
</address>
```

EXPERT NETWORK

Exercise 5: Scaffolding the views and implementation of the controller

DISCUSSION:
(View) Model Validation Types

EXPERT NETWORK

FE View Model Validation

```
4 references | 0 changes | 0 authors, 0 changes
public class UserViewModel
{
    3 references | 0 changes | 0 authors, 0 changes
    public string? FirstName { get; set; }
    3 references | 0 changes | 0 authors, 0 changes
    public string? LastName { get; set; }
    3 references | 0 changes | 0 authors, 0 changes
    public string? Email { get; set; }
}
}
```

```
▼<div class="form-group">
    <label class="control-label" for="FirstName">FirstName
    </label>
    <input class="form-control" type="text" id="FirstName"
    name="FirstName" value>
    <span class="text-danger field-validation-valid" data-
    valmsg-for="FirstName" data-valmsg-replace="true"></span>
  </div>
▼<div class="form-group">
    <label class="control-label" for="LastName">LastName
    </label>
    <input class="form-control" type="text" id="LastName"
    name="LastName" value>
    <span class="text-danger field-validation-valid" data-
    valmsg-for="LastName" data-valmsg-replace="true"></span>
  </div>
▼<div class="form-group">
    <label class="control-label" for="Email">Email</label>
    <input class="form-control" type="text" id="Email" name=
    "Email" value>
    <span class="text-danger field-validation-valid" data-
    valmsg-for="Email" data-valmsg-replace="true"></span>
  </div>
```

EXPERT NETWORK

FE View Model Validation

```
4 references | 0 changes | 0 authors, 0 changes
public class UserViewModel
{
    3 references | 0 changes | 0 authors, 0 changes
    public string FirstName { get; set; }
    3 references | 0 changes | 0 authors, 0 changes
    public string LastName { get; set; }
    3 references | 0 changes | 0 authors, 0 changes
    public string Email { get; set; }
}
}
```

```
<div class="form-group">
    <label class="control-label" for="FirstName">FirstName</label>
    <input class="form-control" type="text" data-val="true" data-val-required="The FirstName field is required."
    id="FirstName" name="FirstName" value>
    <span class="text-danger field-validation-valid" data-valmsg-for="FirstName" data-valmsg-replace="true">
    </span>
</div>
<div class="form-group">
    <label class="control-label" for="LastName">LastName</label>
    <input class="form-control" type="text" data-val="true" data-val-required="The LastName field is required."
    id="LastName" name="LastName" value>
    <span class="text-danger field-validation-valid" data-valmsg-for="LastName" data-valmsg-replace="true">
    </span>
</div>
<div class="form-group">
    <label class="control-label" for="Email">Email</label>
    <input class="form-control" type="text" data-val="true" data-val-required="The Email field is required." id=
    "Email" name="Email" value>
    <span class="text-danger field-validation-valid" data-valmsg-for="Email" data-valmsg-replace="true"></span>
</div>
```

EXPERT NETWORK

FE View Model Validation

```
4 references | 0 changes | 0 authors, 0 changes
public class UserViewModel
{
    [MaxLength(20, ErrorMessage = "Length should be less than 20 characters")]
    3 references | 0 changes | 0 authors, 0 changes
    public string FirstName { get; set; }
    [MaxLength(15, ErrorMessage = "Length should be less than 15 characters")]
    3 references | 0 changes | 0 authors, 0 changes
    public string LastName { get; set; }
    [EmailAddress(ErrorMessage = "Format not recognized")]
    3 references | 0 changes | 0 authors, 0 changes
    public string Email { get; set; }
}
```

```
▼<div class="form-group">
   <label class="control-label" for="FirstName">FirstName</label>
   <input class="form-control" type="text" data-val="true" data-val-maxlength="Length should be less than 20 characters" data-val-maxlength-max="20" data-val-required="The FirstName field is require
   d." id="FirstName" maxlength="20" name="FirstName" value>
   <span class="text-danger field-validation-valid" data-valmsg-for="FirstName" data-valmsg-replace="true"></span>
 </div>
▼<div class="form-group">
   <label class="control-label" for="LastName">LastName</label>
   <input class="form-control" type="text" data-val="true" data-val-maxlength="Length should be less than 15 characters" data-val-maxlength-max="15" data-val-required="The LastName field is required."
   id="LastName" maxlength="15" name="LastName" value>
   <span class="text-danger field-validation-valid" data-valmsg-for="LastName" data-valmsg-replace="true"></span>
 </div>
▼<div class="form-group">
   <label class="control-label" for="Email">Email</label>
   <input class="form-control" type="email" data-val="true" data-val-email="Format not recognized" data-val-required="The Email field is required." id="Email" name="Email" value>
   <span class="text-danger field-validation-valid" data-valmsg-for="Email" data-valmsg-replace="true"></span>
 </div>
```

BE View Model Validation

```
public async Task<IActionResult> Create(Movie movie)
{
    if (!ModelState.IsValid)
    {
        return View(movie);
    }

    _context.Movies.Add(movie);
    await _context.SaveChangesAsync();

    return RedirectToAction(nameof(Index));
}
```

EXPERT NETWORK

Exercise 6: Frontend & Backend View Model Validation