# 1. What are the responsibilities of each layer of the MVC architecture and how are they connected?

Models - Classes that represent the data of the app. The model classes use validation logic to enforce business rules for that data. Typically, model objects retrieve and store model state in a database.

Views - Views are the components that display the app's user interface (UI). Only displays information.

Controllers - Classes that:

- Handle browser requests.
- Retrieve model data.
- Call view templates that return a response.
- Handles and responds to user input and interaction.

Connection:
The controller receives input, interacts with the model to retrieve or update data, and selects a view to render, passing the data from the model to the view for display.

# 2. What are the naming conventions for models, controllers, controller actions, views folders and views themselves?

*All files names in format where every word in concatenation starts from UpperCase:*

- **Models**: located in the /Models folder (e.g., Movie.cs).
- **Controllers**: Located in the /Controllers folder. Each controller is a class ending with Controller (e.g., MoviesController.cs).
- **Controller Actions**: Public methods inside controllers, (e.g., Index, Edit, Details).
- **View Folders**: Located in /Views, with a subfolder matching the controller name (e.g., /Views/Movies/ for MoviesController).
- **View Files**: named after the corresponding action method (e.g., Index.cshtml, Edit.cshtml).

## 3. How to pass data from controllers to views (2 options)?

1. Do this by having the controller put the dynamic data (parameters) that the view template needs in a **ViewData** dictionary. Use:

```
ViewData["Key"] = value;
```

2. **ViewModel** - create a model class and pass it to the view using:

```
return View(model);
```

In the view, declare the model at the top using:

```
@model YourModelType
```

## 4. How to map URL's to controller actions?

The MVC model biding system automatically maps the named parameters from the query string to parameters in the method based off route pattern which in this case is:

```
pattern: "{controller=Home}/{action=Index}/{id?}")
```

Example:

/Movies/Edit/5

maps to the

Edit action in MoviesController, with id = 5.

## 5. How to restrict controller actions to be executed only via certain HTTP request types (e.g., only via POST)?

By adding the [Http<RequestName>] header for specific controller actions. E.g the HttpPost attribute specifies that e.g Edit method can be invoked only for POST requests.

## 6. How to make sure a controller action can only be called through a form on our website and not through some external request?

Add the header to the controller action [ValidateAntiForgeryToken].

## 7. Where do you define data validation and how do you ensure it in views and controllers?

You can declaratively specify validation rules in one place (in the model class) and the rules are enforced everywhere in the app.

The DataAnnotations namespace provides a set of built-in validation attributes that are applied declaratively to a class or property, e.g.:
[Required]
[Range(1,100)]
[DataType(DataType.Currency)]
[RegularExpression()]
[StringLength(5)]

The controller method calls ModelState.IsValid to check whether there are any validation errors. Calling this method evaluates any validation attributes that have been applied to the object