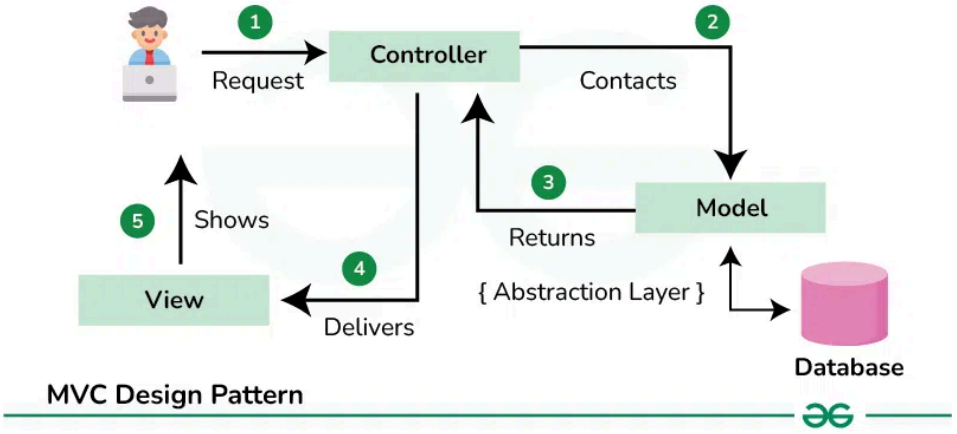
1. What are the responsibilities of each layer of the MVC architecture and how are they connected?
Model:
Handles the data and business logic. It retrieves, stores, and manages the data typically through a database.
Connected to: Controllers

View:
Displays data to the user and collects user input. It's the UI layer using Razor syntax.
Connected to: Controllers (receives data via ViewData, ViewBag).

Controller:
Processes incoming requests, performs operations using the model, and returns a response (usually a view).
Connected to: Views and Models.



Source: https://www.geeksforgeeks.org/mvc-design-pattern/

2. What are the naming conventions for models, controllers, controller actions, views folders, and views themselves?
Models:
Singular form with PascalCase (e.g., Movie.cs).

Controllers:
PascalCase, end with "Controller" (e.g., HomeController.cs, MoviesController.cs).

Controller Actions:
PascalCase methods inside controllers (e.g., Index, Create).

Views Folder:
Located in the Views directory; each controller has its own subfolder named after the controller (minus "Controller" suffix), e.g., Views/Movies/.

Views:
Named after the action method (e.g., Index.cshtml, Create.cshtml).

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

1. Using a Model object – Pass strongly-typed data with return View(model);
2. Using ViewData or ViewBag – Key-value pairs:
    - ViewData["Message"] = "Hello";
    - ViewBag.Message = "Hello";

4. How to map URLs to controller actions?

ASP.NET MVC uses routing, defined in Program.cs or Startup.cs. Default route pattern:

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

URL /Movies/Details/5 maps to MoviesController.Details(int id = 5)

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

Use HTTP method attributes above the action method:

- [HttpPost]
  public IActionResult Create(Movie movie) { ... }
- [HttpGet]
  public IActionResult Create() { ... }

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

Use [ValidateAntiForgeryToken] with an anti-forgery token in the form:

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Movie movie) { ... }
```

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

- Define in the Model using Data Annotations:
  ```
  public class Movie {
     [StringLength(60, MinimumLength = 3)]
     [Required]
     public string Title { get; set; }
  }
  ```

- In Controller:
  ```
  if (ModelState.IsValid) {
     // Save and redirect
  }
  ```

- In View: Add asp-validation-for and include validation scripts:
  ```
  <label asp-for="Title" class="control-label"></label>
  <input asp-for="Title" class="form-control" />
  <span asp-validation-for="Title" class="text-danger"></span>
  ```