# Module 2

Designing ASP.NET Core MVC Web Applications

# Module Overview

- Planning in the Project Design Phase
- Designing Models, Controllers and Views

# Lesson 1: Planning in the Project Design Phase

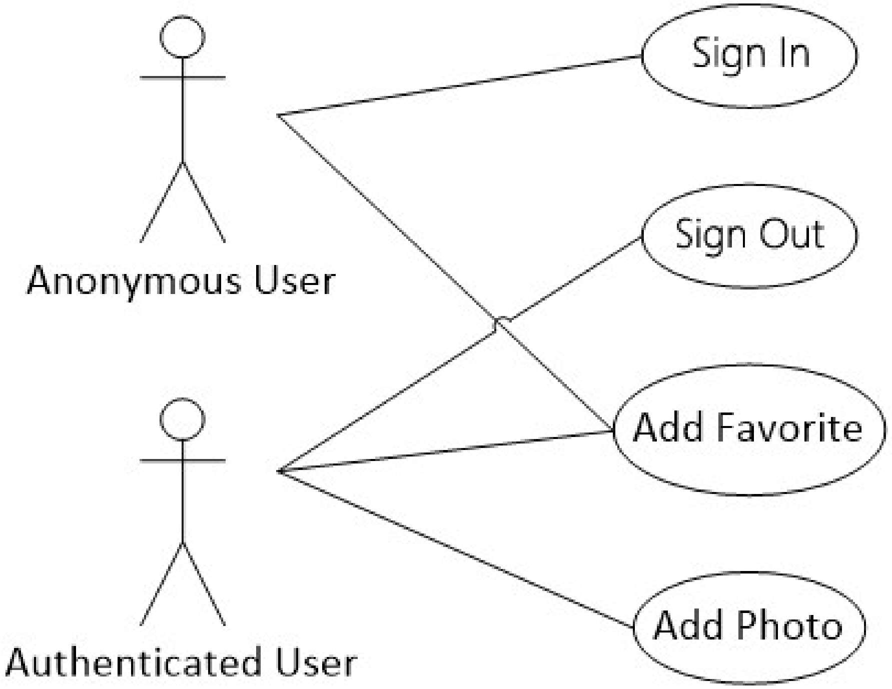
- Project Development Methodologies
- Gathering Requirements
- Planning the Database Design
- Planning for Distributed Applications
- Planning State Management
- Planning Globalization and Localization
- Planning Accessible Web Applications

# Project Development Methodologies

| **Development Model** | **Description** |
|---|---|
| Waterfall Model | Activities for building an application are performed sequentially in distinct phases with clear deliverables. |
| Iterative Development Model | Activities for building an application are performed iteratively in parts by using working versions that are thoroughly tested, until it is finalized. |
| Prototype Model | Activities for building an application are based on a few business requirements, and a prototype is made. Feedback on the prototype is used as an input to develop the final application. |
| Agile Development Model | Activities for building an application are performed in rapid cycles, integrating changing circumstances and requirements in the development process. |
| Extreme Programming | Activities for building an application begin with solving a few critical tasks. Developers test the simplified solution and obtain feedback from stakeholders to derive the detailed requirements, which evolve over the project life cycle. |
| Test Driven Development | Activities for building an application begin with a test project. Changes to the code can be tested singly or as a group, throughout the project. |
| Unified Modeling Language | Activities for building an application begin with UML diagrams that are used for planning and documenting purposes, across all project development models. |

# Gathering Requirements

- Functional requirements describe how the application responds to users
- Technical requirements describe the technical features of an application, such as availability, security, or performance
- You can build functional requirements by using:
  - Usage scenarios
  - Use cases
  - Requirements modeling in the agile methodology
  - User stories in the extreme programming methodology



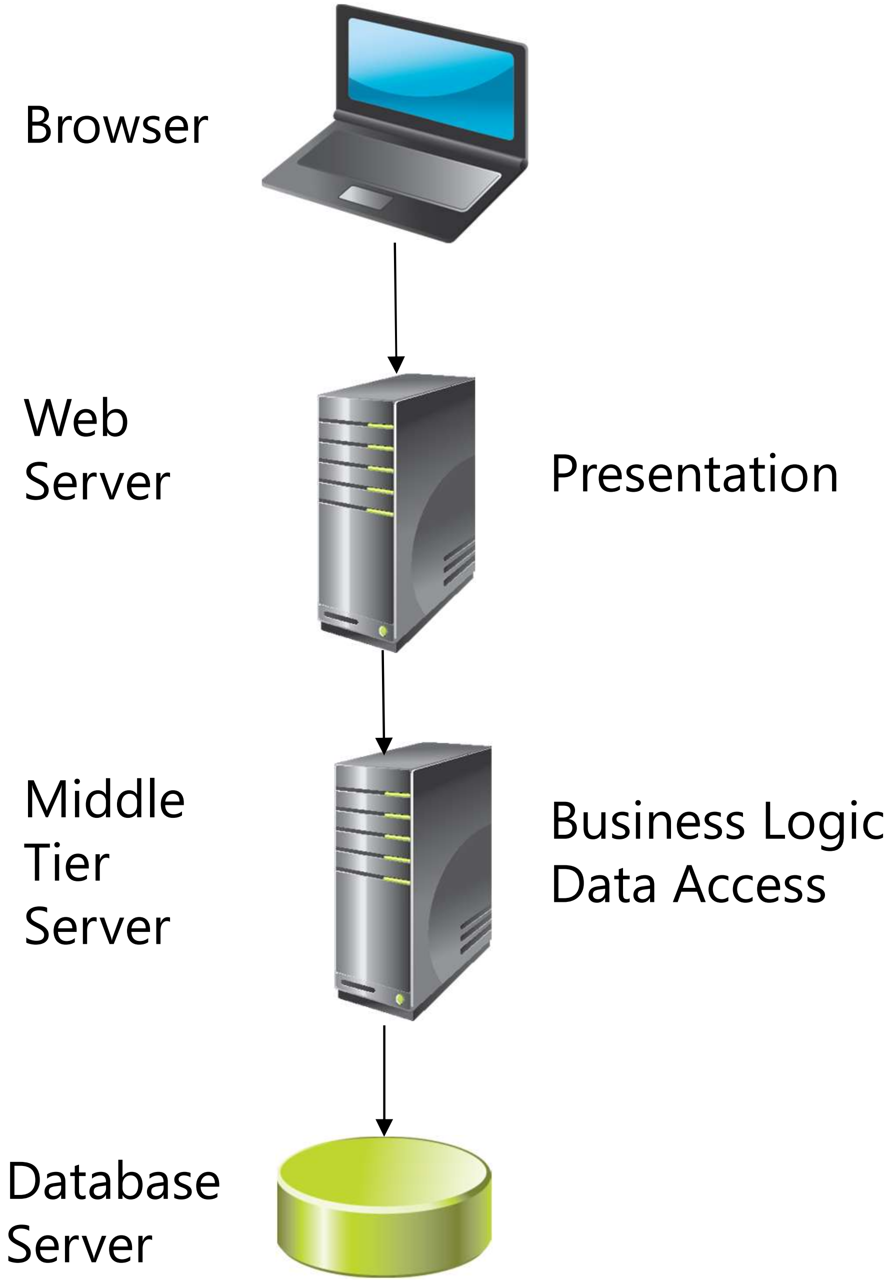
Sample UML Use Case Diagram

# Planning the Database Design

- Logical modeling
- Physical database structure
- Working with DBAs
- Database design in agile and extreme programming

# Planning for Distributed Applications

- Layers
  - Presentation
  - Business logic
  - Data access
  - Database
- Communication
- Security



# Planning State Management

- Client-side locations to store state data:
  - Cookies
  - Query strings
- Server-side locations to store state data:
  - TempData
  - Application state
  - Session state
  - Database tables

# Planning Globalization and Localization

- You can use the internationally-recognized set of language codes available in browsers to present content customized to suit a user’s language or region
- You can use resource files to provide a localized response suitable to a user’s culture
- You can use separate views to suit each language code

# Planning Accessible Web Applications

You can ensure that your content is accessible to the broadest range of users by adhering to the following guidelines:

- Provide **alt** attributes for visual and auditory content
- Do not rely on color to highlight content
- Separate content from structure and presentation code:
  - Only use tables to present tabular content
  - Avoid nested tables
  - Use **<div>** elements and positional style sheets to lay out elements on the page
  - Avoid using images that include important text
  - Put all important text in HTML elements or **alt** attributes

# Lesson 2: Designing Models, Controllers and Views

- Designing Models
- Designing Controllers
- Designing Views
- Information Architecture

# Designing Models

- Model classes and properties
- Using diagrams
- Relationships and aggregates
- Entity framework
- Design in agile and extreme programming

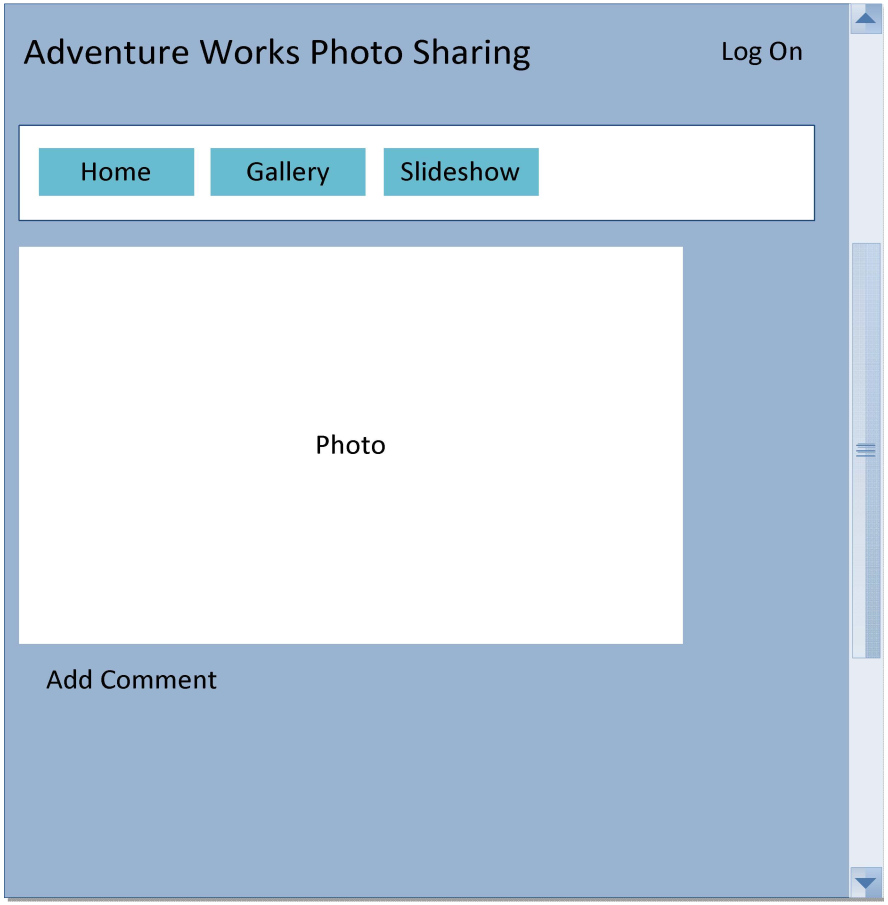
# Designing Controllers

| Controller | Action |
|---|---|
| Photo | AddPhoto (GET) |
| | AddPhoto (POST) |
| | DisplayGallery (GET) |
| User | Logon (GET) |
| | Logon (POST) |

- Identify controllers and actions
- Design in agile and extreme programming

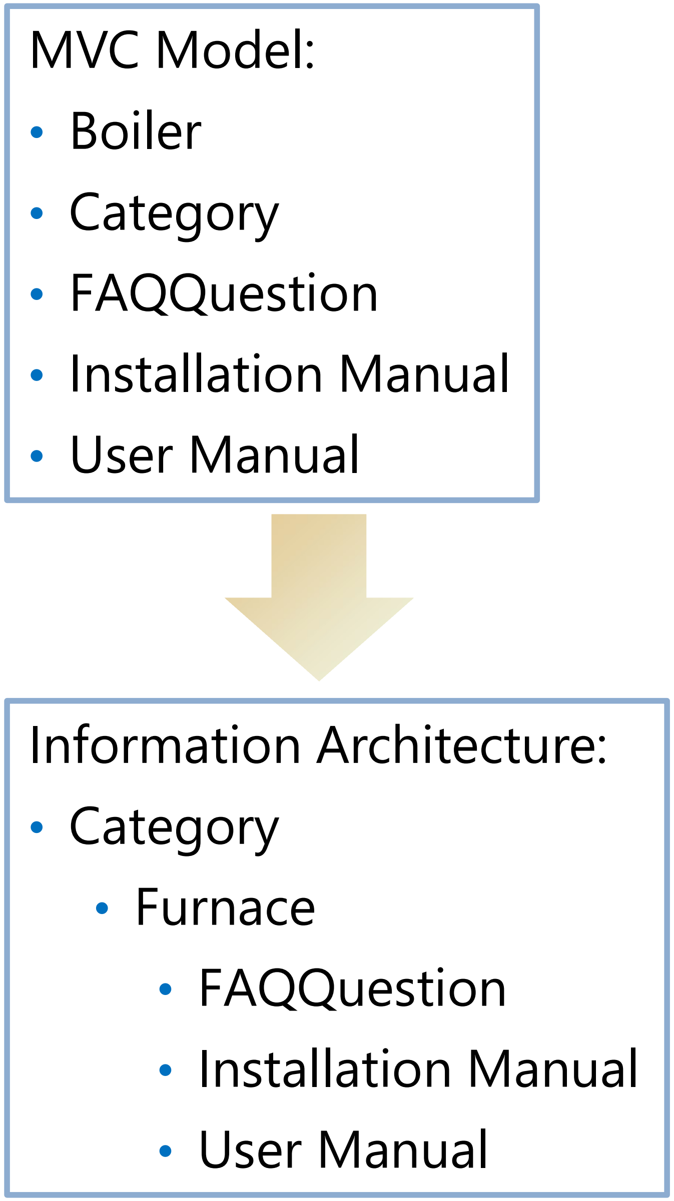
# Designing Views

- Views
- Layouts
- Partial views and view components
- Design in agile and extreme programming



# Information Architecture

- Planning a logical hierarchy
- Presenting a hierarchy in navigation controls
- Presenting a hierarchy in URLs



MVC Model:

- Boiler
- Category
- FAQQuestion
- Installation Manual
- User Manual

Information Architecture:

- Category
  - Furnace
    - FAQQuestion
    - Installation Manual
    - User Manual

# Lab: Designing ASP.NET Core MVC Web Applications

- Exercise 1: Planning Model Classes
- Exercise 2: Planning Controllers
- Exercise 3: Planning Views
- Exercise 4: Architecting an MVC Web Application

Estimated Time: 45 minutes

# Lab Scenario

Your team has chosen ASP.NET Core MVC as the most appropriate ASP.NET programming model to create the photo sharing application for the Adventure Works web application. You need to create a detailed project design for the application and have been given a set of functional and technical requirements with other information. You have to plan:

- An MVC model that you can use to implement the desired functionality.
- One or more controllers and controller actions that respond to users actions.
- A set of views to implement the user interface.
- The locations for hosting and data storage.

# Lab Review

- What model classes should be created for the photo sharing application based on the initial investigation?
- What controllers should be created for the photo sharing application based on the initial investigation?
- What views should be created for the photo sharing application?

# Module Review and Takeaways

- Review Question
- Real-world Issues and Scenarios
- Tools
- Best Practice
- Common Issues and Troubleshooting Tips