Moving around

Hold spacebar, then click and drag to move around.

Find it

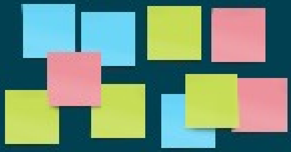
Hold **spacebar,** then click and drag to move around.

See it

Try it

① Hold **spacebar** until you see a hand.

② Place your **cursor** over the stickies, then **click and drag.**

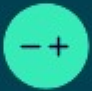


Edit the board when your cursor looks like a **pointer** (v). Move around with the **hand** (h).

Zooming in and out

Use the plus and minus keys on your keyboard to zoom in and out.

Find it



Use the **plus** and **minus** keys on your keyboard.

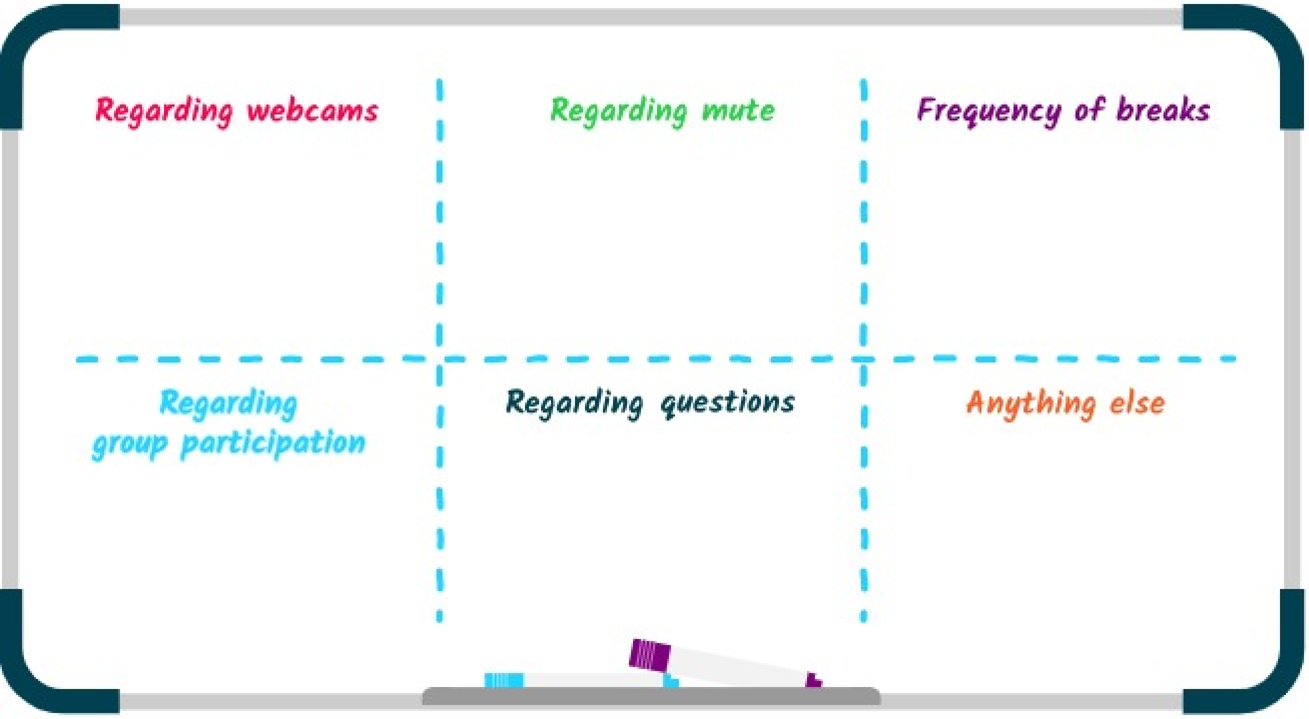
See it

Try it

① Center your screen on the small text below.

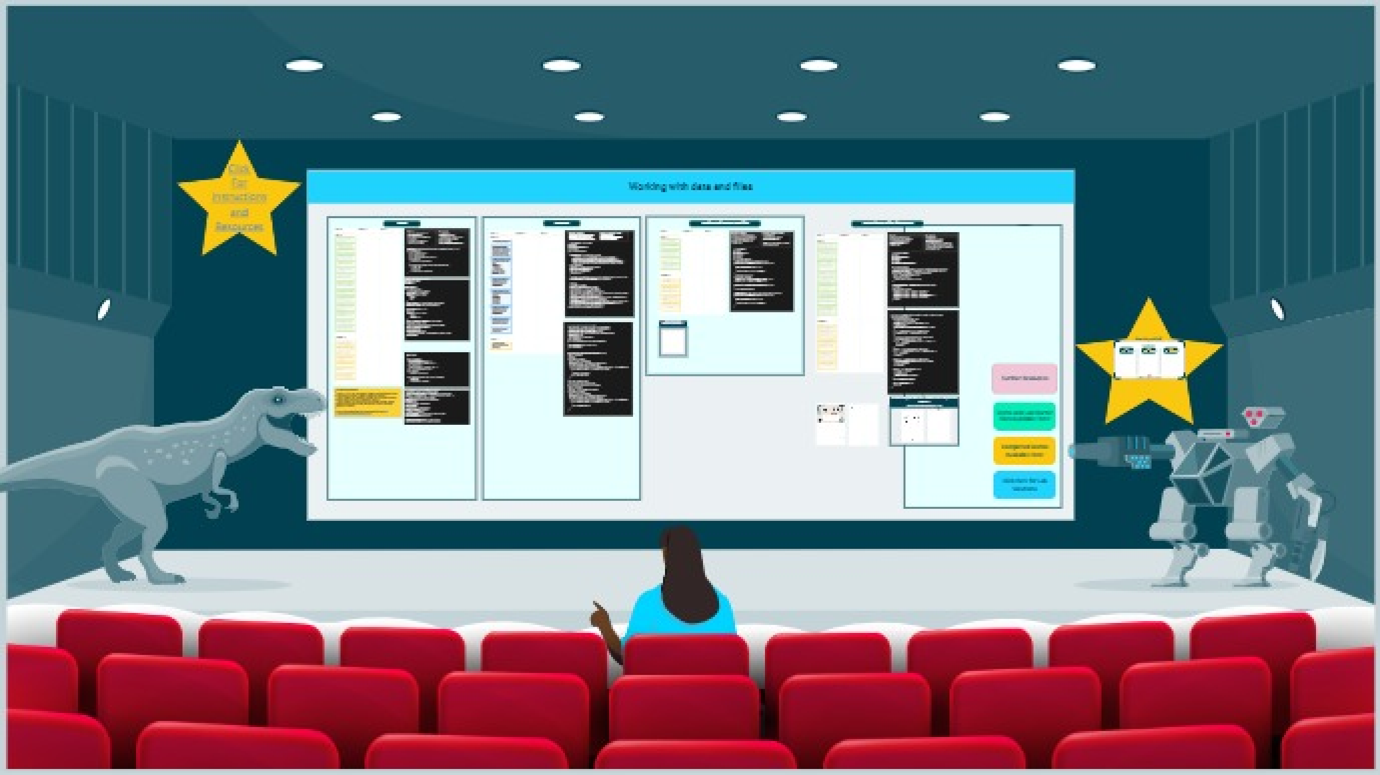
② Use the **plus** and **minus** keys to zoom in and read the message.

Alternatively, use the **scroll wheel** on a mouse or **pinch** on a trackpad.



Timings

Start	9:30	
Coffee	15 mins	Starts anytime between 10:45 and 11:00
Lunch	60 mins	Starts anytime between 12:30 and 13:00
Tea	15 mins	Starts anytime between 15:00 and 15:15
End	16:30 - 16:45	


Click For Instructions and Resources
Working with data and files

JSON

To Do | 20

In progress | 0

Done | 0

Topics | 14

1. Introduce topic: what JSON is; why it's useful; how it relates to C# classes and objects.

2. Set up solution.

3. Set up new console project: json-demo and add to solution.

4. Create PersonFilms class (Name, Age and FavouriteFilms properties; default + specific constructors).

5. Create PersonFilms object with values.

6. Serialize personFilm object to string; display string.

7. Deserialize JSON string back into new PersonFilms object; display contents using strongly typed.

8. Use JsonProperty attributes to control the mapping of property names to JSON names (name, age-in

9. Show serialization now different, but still mapped back to

10. Show deserialization to dynamic object, then access via property name or [name] syntax (age-in

11. Show serialization of anonymous object (no control over mapping one-to-one direct).

12. Show deserialization of unrecognized elements is ignored (not an error).

13. Show deserialization of missing elements (Age - defaults to zero).

14. Demonstrate use of JsonProperty attributes to control the mapping of property names to JSON names.

+

Lab activities | 6

1. Write a class suitable for holding film information (film_id, title, synopsis, director, release_date).

2. Deserialize a JSON file containing films into a List<Film> object. Use the JSON listed below on the yellow background to create the file content.

3. Display the film information on a console screen.

4. Add another film to the list.

5. Serialize the list back out to JSON file again.

6. If get time, duplicate code but using dynamic / anonymous types rather than purpose built classes.

+

// JSON file contents for lab step 2
[
{ "film_id": "112233", "title": "King of Thieves", "synopsis": "King Charles gets mixed up with a bunch of wrong 'uns", "director": "H. Windsor", "release_date": "2019-05-27" },
{ "film_id": "445566", "title": "The Predator", "synopsis": "Felix the cat gets into some heavy mousing", "director": "Fifi La Chatte", "release_date": "2019-10-19" },
{ "film_id": "999999", "title": "The House with a Clock in its Walls", "synopsis": "A builder accidently leaves a clock inside the walls of his latest project", "director": "Bob Builder", "release_date": "2018-07-03" }
]

You can download the file (Films.json) that contains the above data from [QACS-TL/WorkingWithDataAndFilesStarterFiles (github.com)]

Visual Studio

```
//From within VS (using point and click)
// Create new solution called
//WorkingWithDataAndFiles

//Add console project called json-demo.csproj
//to the solution

// Use NuGet to add a reference to the
// Newtonsoft.json JSON package
```

Visual Studio Code

```
//From Command Prompt
// Prepare solution
dotnet new solution --name demo

// Create console project, and add to solution
dotnet new console --output json-demo
dotnet sln add json-demo

// Install JSON package in the json-demo project
dotnet add json-demo package newtonsoft.json
```

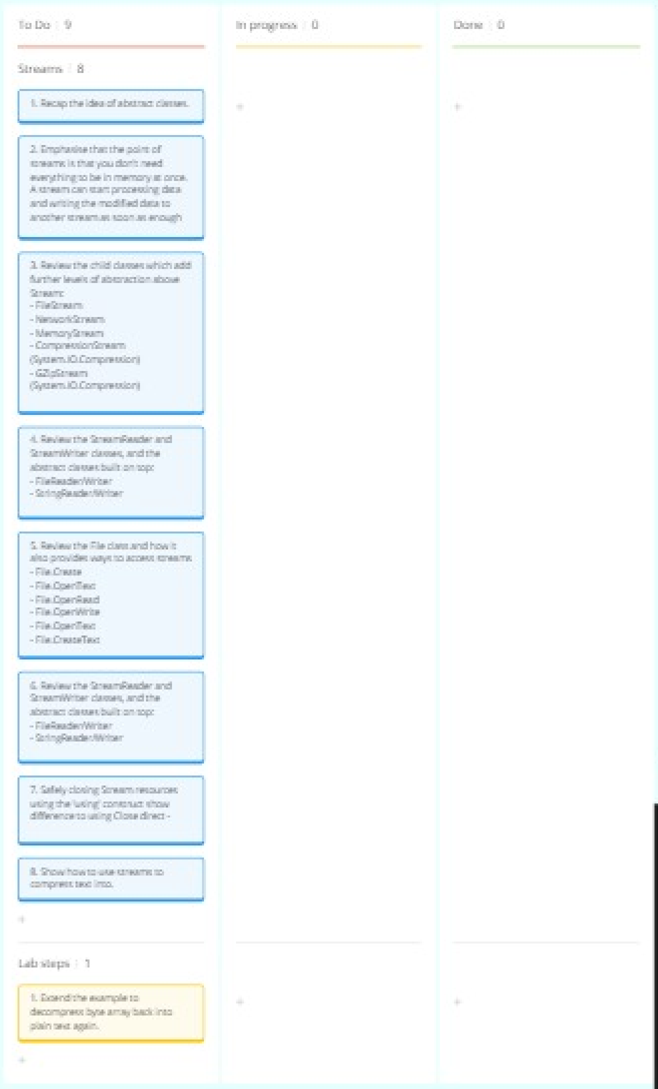
```
// PersonFilms class (name, age and FavouriteFilms) - NB: need default constructor for JSON
deserialization
public class PersonFilms
{
    public string Name { get; set; }
    public int Age { get; set; }
    public List<string> FavouriteFilms { get; set; }

    public PersonFilms()
    {
        FavouriteFilms = new List<string>();
    }

    public PersonFilms(string name, int age, List<string> favouriteFilms)
    {
        this.Name = name;
        this.Age = age;
        this.FavouriteFilms = favouriteFilms;
    }
}
```

```
// Include all relevant namespaces including json
using System;
using System.Collections.Generic;
using System.IO;
using Newtonsoft.Json;

namespace json_demo{
  internal class Program {
  static void Main(string[] args)
  {
    // Writing and reading a strongly typed class
    PersonFilms personFilms = new PersonFilms {
      Name = "Andrew",
      Age = 58,
      FavouriteFilms = new  List<string>() {
          "Toy Story",
          "Toy Story 2",
          "Aliens"
      }
    };
    string pfs = JsonConvert.SerializeObject(personFilms, Formatting.Indented);
    Console.WriteLine(pfs);
    PersonFilms pfs2 = JsonConvert.DeserializeObject<PersonFilms>(pfs);
    Console.WriteLine(pfs2.Name);

    // Writing an anonymous object to file
    var obj1 = new
    {
      name = "Doris",
      age = 29,
      favouriteFilms = new  List<string>() {
          "Love Story",
          "Inception",
          "It's a Wonderful Life"
      }
    };
    string s1 = JsonConvert.SerializeObject(obj1);
    string json1 = JsonConvert.SerializeObject(obj1, Formatting.Indented);
    File.WriteAllText("file1.json", json1);

    // Reading JSON into an anonymous, dynamic object then picking out elements
    string s2 = File.ReadAllText("file1.json");
    dynamic obj2 = JsonConvert.DeserializeObject(s2);
    Console.WriteLine(obj2.name);
    Console.WriteLine(obj2.age);
    Console.WriteLine(obj2.favouriteFilms);

    // Alternative way to read elements, if they are non-standard (include dots or hyphens)
    int age = obj2["age"];
    Console.WriteLine(age);
  }
}
}
```

```
  using Newtonsoft.Json;

  namespace json_demo
  {
    internal class PersonFilms2
    {
      [JsonProperty(PropertyName = "name", Order = 2)]
      public string Name { get; set; }

      [JsonProperty(PropertyName = "age-in-years", Order = 1)]
      public int Age { get; set; }

      [JsonProperty(PropertyName = "favourite-films", Order = 3)]
      public List<string> FavouriteFilms { get; set; }

      public PersonFilms2()
      {
          FavouriteFilms = new List<string>();
      }

      public PersonFilms2(string name, int age, List<String> favouriteFilms)
      {
          this.Name = name;
          this.Age = age;
          this.FavouriteFilms = favouriteFilms;
      }
    }
  }
```

```
ADD THE FOLLOWING CODE TO THE FOOT OF THE PROGRAM CLASS'S MAIN METHOD

// Use JsonProperty attributes to control the mapping of property
 // names to JSON names (name, age-in-years, favourite-films)
 // Note serialization order specified in class via the JsonPropertyAttribute Order property
 PersonFilms2 pf2 = new PersonFilms2("Sharon", 27 ,new List<string>{ "Top Gun", "Die Hard", "ET"});
 s3 = JsonConvert.SerializeObject(pf2, Formatting.Indented);

 Console.WriteLine(s3);
 PersonFilms2 pd2 = JsonConvert.DeserializeObject<PersonFilms2>(s3);
 string name = pd2.Name;
 age = pd2.Age;
 List<string> favouriteFilms = pd2.FavouriteFilms;
 Console.WriteLine(name);
 Console.WriteLine(age);
 favouriteFilms.ForEach(f => Console.WriteLine(f));

 dynamic pd3 = JsonConvert.DeserializeObject(s3);
 string nameAA = pd3["name"];
 int ageInYearsAA = pd3["age-in-years"];
 List<string> favouriteFilmsAA = pd3["favourite-films"].ToObject<List<string>>();
 Console.WriteLine(nameAA);
 Console.WriteLine(ageInYearsAA);
 favouriteFilmsAA.ForEach(f => Console.WriteLine(f));

 // Deserialization of unrecognized elements is ignored (not an error)
 //Console.WriteLine(obj2.ages);
 Console.WriteLine($"{(obj2.ages != null ? obj2.ages : string.Empty)}");
```

Streams



To Do | 9

Streams | 8

1. Recap the idea of abstract classes.

2. Emphasise that the point of streams is that you don't need everything to be in memory at once. A stream can start processing data and writing the modified data to another stream as soon as enough

3. Review the child classes which add further levels of abstraction above Stream:
- FileStream
- NetworkStream
- MemoryStream
- CompressionStream (System.IO.Compression)
- GZipStream (System.IO.Compression)

4. Review the StreamReader and StreamWriter classes, and the abstract classes built on top:
- FileReader/Writer
- StringReader/Writer

5. Review the File class and how it also provides ways to access streams:
- File.Create
- File.OpenText
- File.OpenRead
- File.OpenWrite
- File.OpenText
- File.CreateText

6. Review the StreamReader and StreamWriter classes, and the abstract classes built on top:
- FileReader/Writer
- StringReader/Writer

7. Safely closing Stream resources using the 'using' construct show difference to using Close direct -

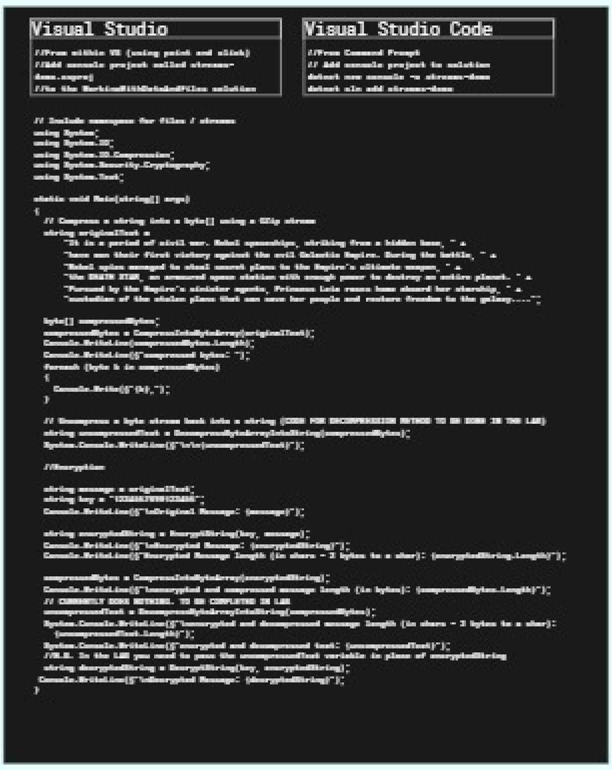
8. Show how to use streams to compress text into.

Lab steps | 1

1. Extend the example to decompress byte array back into plain text again.

In progress | 0

Done | 0



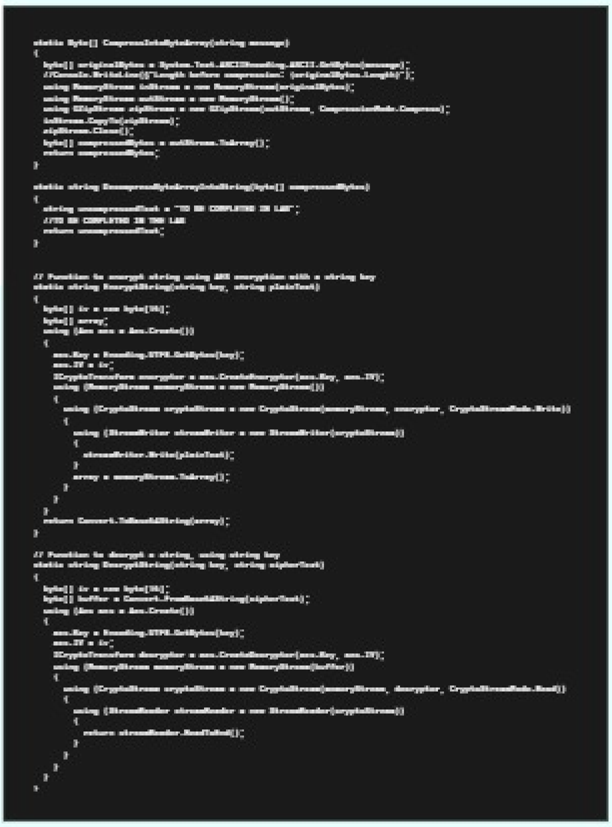
Visual Studio

```
//From within VS (using point and click)
//Add console project called streams-demo.csproj
//to the WorkingWithDataAndFiles solution
```

Visual Studio Code

```
//From Command Prompt
// Add console project to solution
dotnet new console -o streams-demo
dotnet sln add streams-demo
```

```
// Include namespace for files / streams
using System;
using System.IO;
using System.IO.Compression;
using System.Security.Cryptography;
using System.Text;

static void Main(string[] args)
{
  // Compress a string into a byte[] using a GZip stream
  string originalText =
      "It is a period of civil war. Rebel spaceships, striking from a hidden base, " +
      "have won their first victory against the evil Galactic Empire. During the battle, " +
      "Rebel spies managed to steal secret plans to the Empire's ultimate weapon, " +
      "the DEATH STAR, an armored space station with enough power to destroy an entire planet. " +
      "Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, " +
      "custodian of the stolen plans that can save her people and restore freedom to the galaxy....";

  byte[] compressedBytes;
  compressedBytes = CompressIntoByteArray(originalText);
  Console.WriteLine(compressedBytes.Length);
  Console.WriteLine($"compressed bytes: ");
  foreach (byte b in compressedBytes)
  {
    Console.Write($"{b},");
  }

  // Decompress a byte stream back into a string (CODE FOR DECOMPRESSION METHOD TO BE DONE IN THE LAB)
  string uncompressedText = DecompressByteArrayIntoString(compressedBytes);
  System.Console.WriteLine($"\n\n{uncompressedText}");

  //Encryption

  string message = originalText;
  string key = "1234567890123456";
  Console.WriteLine($"\nOriginal Message: {message}");

  string encryptedString = EncryptString(key, message);
  Console.WriteLine($"\nEncrypted Message: {encryptedString}");
  Console.WriteLine($"Encrypted Message Length (in chars - 2 bytes to a char): {encryptedString.Length}");

  compressedBytes = CompressIntoByteArray(encryptedString);
  Console.WriteLine($"\nencrypted and compressed message length (in bytes): {compressedBytes.Length}");
  // CURRENTLY DOES NOTHING. TO BE COMPLETED IN LAB
  uncompressedText = DecompressByteArrayIntoString(compressedBytes);
  System.Console.WriteLine($"\nencrypted and decompressed message length (in chars - 2 bytes to a char):
    {uncompressedText.Length}");
  System.Console.WriteLine($"encrypted and decompressed text: {uncompressedText}");
  //N.B. In the LAB you need to pass the uncompressedText variable in place of encryptedString
  string decryptedString = DecryptString(key, encryptedString);
 Console.WriteLine($"\nDecrypted Message: {decryptedString}");
}
```



```
static Byte[] CompressIntoByteArray(string message)
{
  byte[] originalBytes = System.Text.ASCIIEncoding.ASCII.GetBytes(message);
  //Console.WriteLine($"Length before compression: {originalBytes.Length}");
  using MemoryStream inStream = new MemoryStream(originalBytes);
  using MemoryStream outStream = new MemoryStream();
  using GZipStream zipStream = new GZipStream(outStream, CompressionMode.Compress);
  inStream.CopyTo(zipStream);
  zipStream.Close();
  byte[] compressedBytes = outStream.ToArray();
  return compressedBytes;
}

static string DecompressByteArrayIntoString(byte[] compressedBytes)
{
  string uncompressedText = "TO BE COMPLETED IN LAB";
  //TO BE COMPLETED IN THE LAB
  return uncompressedText;
}


// Function to encrypt string using AES encryption with a string key
static string EncryptString(string key, string plainText)
{
  byte[] iv = new byte[16];
  byte[] array;
  using (Aes aes = Aes.Create())
  {
    aes.Key = Encoding.UTF8.GetBytes(key);
    aes.IV = iv;
    ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
    using (MemoryStream memoryStream = new MemoryStream())
    {
      using (CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor, CryptoStreamMode.Write))
      {
        using (StreamWriter streamWriter = new StreamWriter(cryptoStream))
        {
          streamWriter.Write(plainText);
        }
        array = memoryStream.ToArray();
      }
    }
  }
  return Convert.ToBase64String(array);
}

// Function to decrypt a string, using string key
static string DecryptString(string key, string cipherText)
{
  byte[] iv = new byte[16];
  byte[] buffer = Convert.FromBase64String(cipherText);
  using (Aes aes = Aes.Create())
  {
    aes.Key = Encoding.UTF8.GetBytes(key);
    aes.IV = iv;
    ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);
    using (MemoryStream memoryStream = new MemoryStream(buffer))
    {
      using (CryptoStream cryptoStream = new CryptoStream(memoryStream, decryptor, CryptoStreamMode.Read))
      {
        using (StreamReader streamReader = new StreamReader(cryptoStream))
        {
          return streamReader.ReadToEnd();
        }
      }
    }
  }
}
```

Reading and writing CSV files

To Do | 9

In progress | 0

Done | 0

Topics | 5

1. Recap structure of CSV file, variations, and complexities.

2. Show how to install CSVHelper library.

3. Show using CSV library to read data from CSV file into list, using strongly typed class as mapping.

4. Show how to write list of strongly typed objects to CSV file.

5. Show reading into List<dynamic> without having to define a class first (but everything is a string).

Lab activities | 4

1. Download a copy of "streaming_movies.csv" to your computer (see below for instructions.

2. Write code to read the CSV file "streaming_movies.csv" into a list (like we did in the demo), using a custom class suitable for the data

3. Filter the list to get just movies on the Netflix platform, and sort it by movie title.

4. Write the filtered list of movies to a new CSV file, but only including the columns: Title, ReleaseYear, and

Visual Studio

```
//From within VS (using point and click)
//Add console project called csv-demo.csproj
//to the WorkingWithDetailedFiles solution

//Use NuGet to get a reference to the
//CSVHelper library
```

Visual Studio Code

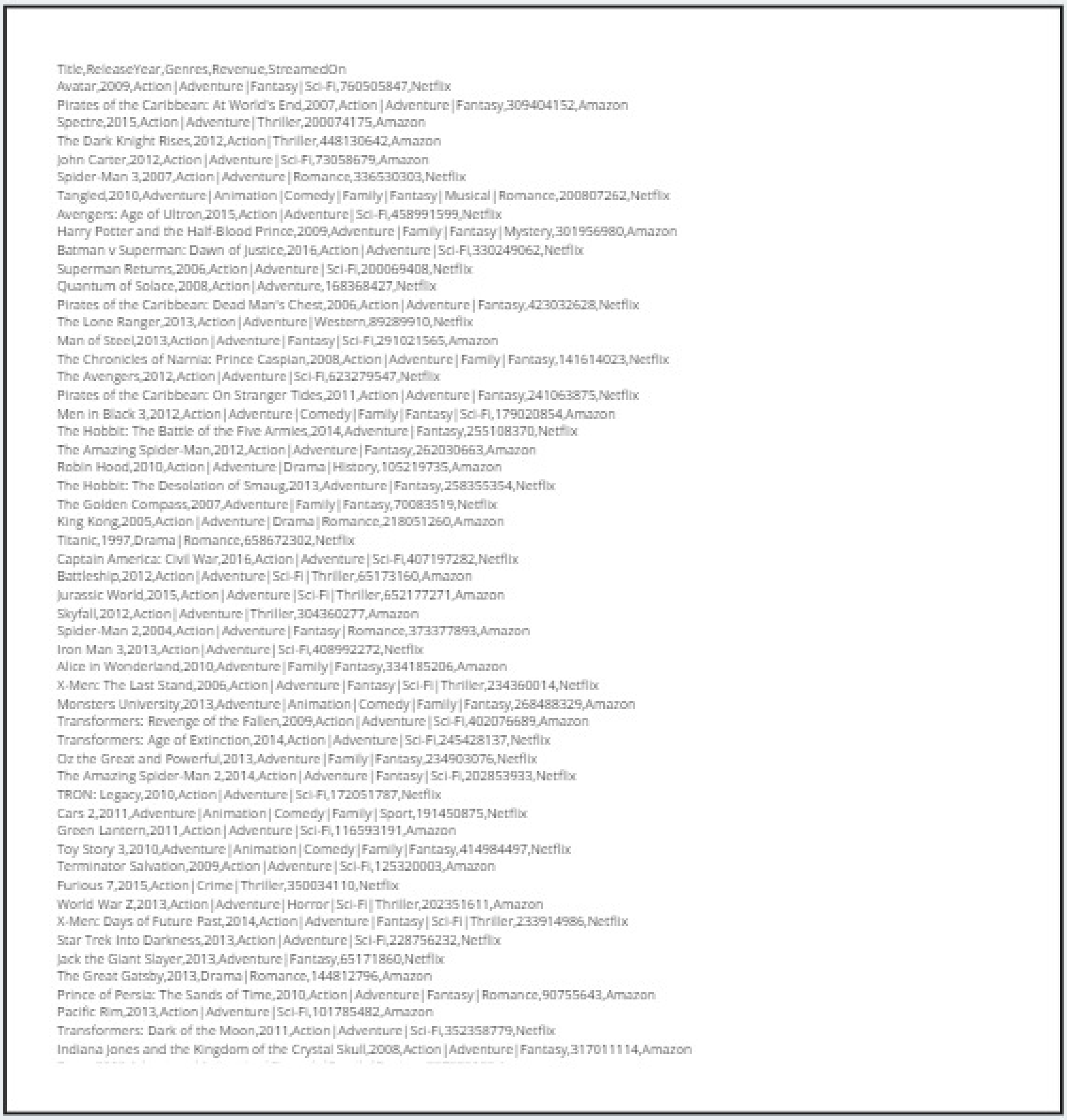
```
//From Command Prompt
// Add console project to solution
dotnet new console -o csv-demo
dotnet sln add csv-demo

//Include the CSVHelper package in the csv-demo project
dotnet add package CsvHelper
```

```
// Include namespaces
using System;
using System.IO;
using System.Globalization;
using System.Linq;
using CsvHelper;

static void Main(string[] args)
{
  // Reading and displaying a list of people from a CSV file
  using (var sr = new StreamReader("movies.csv"))
  using (var reader = new CsvReader(sr, CultureInfo.InvariantCulture))
  {
    var list = reader.GetRecords<Movie>().ToList();
    foreach (Movie m in list)
    {
      Console.WriteLine($"{m.Title} in {m.ReleaseYear}");
    }
  }

  // Writing a list to a CSV file
  var more_movies = new Movie[] {
    new Movie { Title = "2001: A Space Odyssey", ReleaseYear = 1968 },
    new Movie { Title = "Dark Star", ReleaseYear = 1974 },
    new Movie { Title = "The Martian", ReleaseYear = 2015 }
  };

  using (var sw = new StreamWriter("updated_movies.csv"))
  using (var writer = new CsvWriter(sw, CultureInfo.InvariantCulture))
  {
    writer.WriteRecords(more_movies);
  }

  // Reading records into dynamic class (NB: every property value will be a string!)
  Console.WriteLine();
  using (var sr = new StreamReader("movies.csv"))
  using (var reader = new CsvReader(sr, CultureInfo.InvariantCulture))
  {
    var list = reader.GetRecords<dynamic>().ToList();
    foreach (var m in list)
    {
      Console.WriteLine($"{m.Title} in {m.ReleaseYear}");
    }
  }
}
```

streaming_movies.csv for use with Lab.
Create the file and copy and paste the text into it or alternatively you can download the file from: QACI-TL/WorkingWithDetailedFilesStarterFiles (github.com)

[illegible]

streaming_movies.csv for use with Lab.
Create the file and copy and paste the text into it or alternatively you can download the file from QACS-TL/WorkingWithDataAndFilesStarterFiles (github.com)



```
Title,ReleaseYear,Genres,Revenue,StreamedOn
Avatar,2009,Action|Adventure|Fantasy|Sci-Fi,760505847,Netflix
Pirates of the Caribbean: At World's End,2007,Action|Adventure|Fantasy,309404152,Amazon
Spectre,2015,Action|Adventure|Thriller,200074175,Amazon
The Dark Knight Rises,2012,Action|Thriller,448130642,Amazon
John Carter,2012,Action|Adventure|Sci-Fi,73058679,Amazon
Spider-Man 3,2007,Action|Adventure|Romance,336530303,Netflix
Tangled,2010,Adventure|Animation|Comedy|Family|Fantasy|Musical|Romance,200807262,Netflix
Avengers: Age of Ultron,2015,Action|Adventure|Sci-Fi,458991599,Netflix
Harry Potter and the Half-Blood Prince,2009,Adventure|Family|Fantasy|Mystery,301956980,Amazon
Batman v Superman: Dawn of Justice,2016,Action|Adventure|Sci-Fi,330249062,Netflix
Superman Returns,2006,Action|Adventure|Sci-Fi,200069408,Netflix
Quantum of Solace,2008,Action|Adventure,168368427,Netflix
Pirates of the Caribbean: Dead Man's Chest,2006,Action|Adventure|Fantasy,423032628,Netflix
The Lone Ranger,2013,Action|Adventure|Western,89289910,Netflix
Man of Steel,2013,Action|Adventure|Fantasy|Sci-Fi,291021565,Amazon
The Chronicles of Narnia: Prince Caspian,2008,Action|Adventure|Family|Fantasy,141614023,Netflix
The Avengers,2012,Action|Adventure|Sci-Fi,623279547,Netflix
Pirates of the Caribbean: On Stranger Tides,2011,Action|Adventure|Fantasy,241063875,Netflix
Men in Black 3,2012,Action|Adventure|Comedy|Family|Fantasy|Sci-Fi,179020854,Amazon
The Hobbit: The Battle of the Five Armies,2014,Adventure|Fantasy,255108370,Netflix
The Amazing Spider-Man,2012,Action|Adventure|Fantasy,262030663,Amazon
Robin Hood,2010,Action|Adventure|Drama|History,105219735,Amazon
The Hobbit: The Desolation of Smaug,2013,Adventure|Fantasy,258355354,Netflix
The Golden Compass,2007,Adventure|Family|Fantasy,70083519,Netflix
King Kong,2005,Action|Adventure|Drama|Romance,218051260,Amazon
Titanic,1997,Drama|Romance,658672302,Netflix
Captain America: Civil War,2016,Action|Adventure|Sci-Fi,407197282,Netflix
Battleship,2012,Action|Adventure|Sci-Fi|Thriller,65173160,Amazon
Jurassic World,2015,Action|Adventure|Sci-Fi|Thriller,652177271,Amazon
Skyfall,2012,Action|Adventure|Thriller,304360277,Amazon
Spider-Man 2,2004,Action|Adventure|Fantasy|Romance,373377893,Amazon
Iron Man 3,2013,Action|Adventure|Sci-Fi,408992272,Netflix
Alice in Wonderland,2010,Adventure|Family|Fantasy,334185206,Amazon
X-Men: The Last Stand,2006,Action|Adventure|Fantasy|Sci-Fi|Thriller,234360014,Netflix
Monsters University,2013,Adventure|Animation|Comedy|Family|Fantasy,268488329,Amazon
Transformers: Revenge of the Fallen,2009,Action|Adventure|Sci-Fi,402076689,Amazon
Transformers: Age of Extinction,2014,Action|Adventure|Sci-Fi,245428137,Netflix
Oz the Great and Powerful,2013,Adventure|Family|Fantasy,234903076,Netflix
The Amazing Spider-Man 2,2014,Action|Adventure|Fantasy|Sci-Fi,202853933,Netflix
TRON: Legacy,2010,Action|Adventure|Sci-Fi,172051787,Netflix
Cars 2,2011,Adventure|Animation|Comedy|Family|Sport,191450875,Netflix
Green Lantern,2011,Action|Adventure|Sci-Fi,116593191,Amazon
Toy Story 3,2010,Adventure|Animation|Comedy|Family|Fantasy,414984497,Netflix
Terminator Salvation,2009,Action|Adventure|Sci-Fi,125320003,Amazon
Furious 7,2015,Action|Crime|Thriller,350034110,Netflix
World War Z,2013,Action|Adventure|Horror|Sci-Fi|Thriller,202351611,Amazon
X-Men: Days of Future Past,2014,Action|Adventure|Fantasy|Sci-Fi|Thriller,233914986,Netflix
Star Trek Into Darkness,2013,Action|Adventure|Sci-Fi,228756232,Netflix
Jack the Giant Slayer,2013,Adventure|Fantasy,65171860,Netflix
The Great Gatsby,2013,Drama|Romance,144812796,Amazon
Prince of Persia: The Sands of Time,2010,Action|Adventure|Fantasy|Romance,90755643,Amazon
Pacific Rim,2013,Action|Adventure|Sci-Fi,101785482,Amazon
Transformers: Dark of the Moon,2011,Action|Adventure|Sci-Fi,352358779,Netflix
Indiana Jones and the Kingdom of the Crystal Skull,2008,Action|Adventure|Fantasy,317011114,Amazon
```

Manipulating Office documents

To Do | 12

In progress | 0

Done | 0

Topics | 8

1. Explain / show how Office documents are actually stored as complex packages of XML files.
2. Discuss how with a Word document, there is a main package; within that a document; within that a body; and within that paragraphs, runs, and texts (and also tables,
3. We could try to edit the XML directly (via a simple XML library) but it's way too hard; we need something to help which understands the structure of the
4. Simplest approach is to work with the file directly, not actually running the Office application (this means we can process Office document even on operating systems which can't run Office (like Linux, for example). A suitable package to use is
5. Show creating a new demo project, adding to solution, and adding in the package.
6. Add the relevant namespaces.
7. Show how a Word document file can be opened and parsed for paragraphs, with the text of the paragraphs being written out to the
8. Show how a Word document can be manipulated by searching for particular text, then replacing that paragraph with a different run to replace the text within the run, and

Lab activities | 4

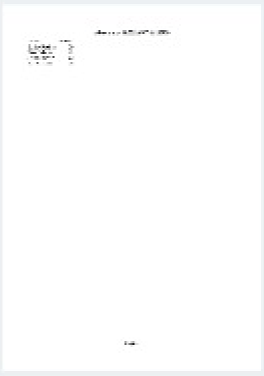
1. Use the 'MovieGiftVoucherTemplate' Word file (downloadable below); it contains a table where the voucher's monetary value needs to be placed in the second row and third column of the table; the name of the recipient needs to be placed in the fourth row and third column of the
2. Ensure you can generate a new copy of the Word file, with the name and value replaced with a specific name and value and the date replaced by a date that is 6 months
3. Extend the project so it can read a CSV file of names and values (downloadable below) and then generate a new voucher document
4. If you have more time, explore and experiment with what else you can do with Office files (including

```
Visual Studio

//From within VS (using point and click)
//Add console project called office-demo.csproj
//to the WorkingWithDelimitedFiles solution

//Use NuGet to get a reference to the
//DocumentFormat.OpenXml and
//CsvHelper libraries
```

```
Visual Studio Code

//From Command Prompt
// Add console project to solution
dotnet new console -o office-demo
dotnet sln add office-demo

//Install OpenXml package in the office-demo project
dotnet add office-demo package DocumentFormat.OpenXml

//Include the CSVHelper package in the csv-demo project
dotnet add package CsvHelper
```

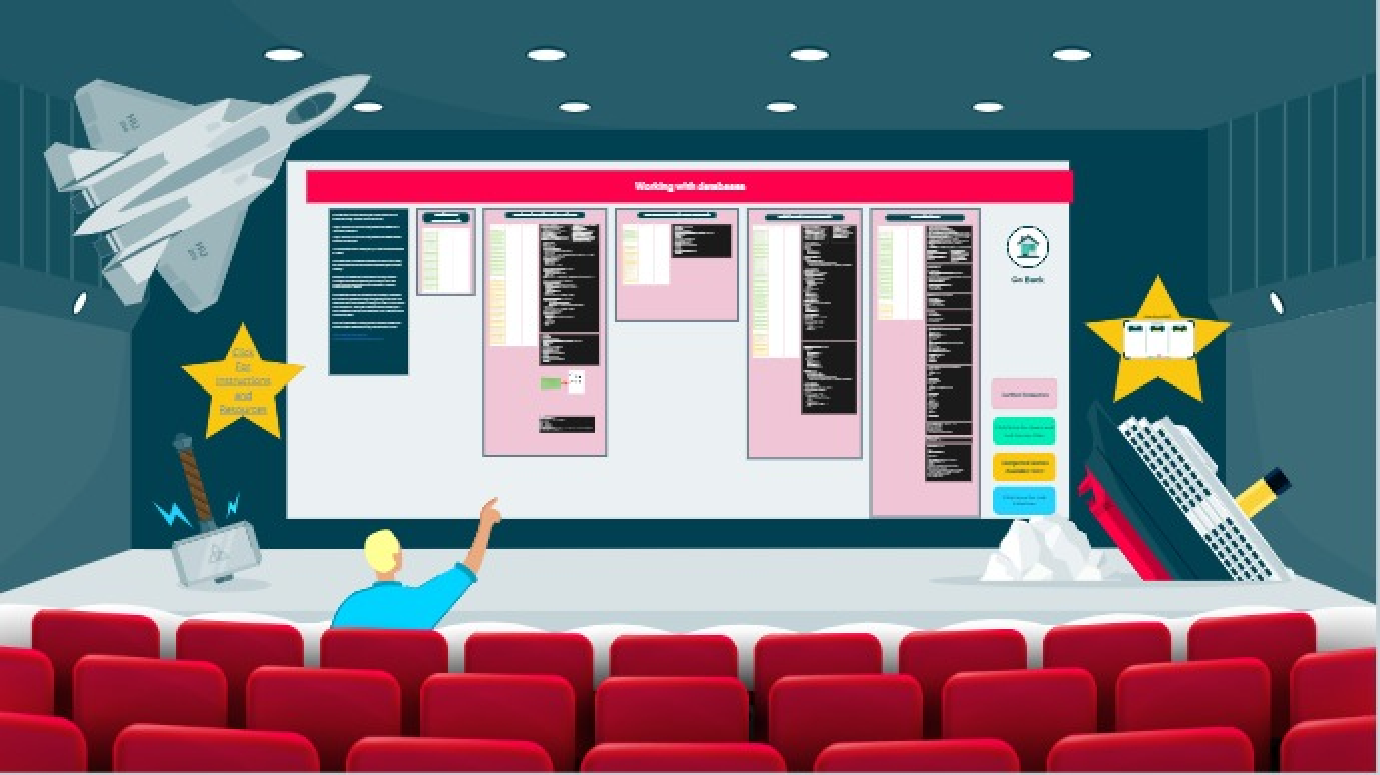
```
// Include relevant namespaces
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using CsvHelper;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;

static void Main(string[] args)
{
  //The following code reads movie details in from a csv file and adds the data into a Word table
  List<Movie> movieList = null;
  using (var sr = new StreamReader("Top revenue earning SciFi movies per alphabetic letter.csv"))
  using (var reader = new CsvReader(sr, CultureInfo.InvariantCulture))
  {
    movieList = reader.GetRecords<Movie>().ToList();
    foreach (Movie m in movieList)
    {
      Console.WriteLine($"{m.Movie_ID} is {m.Title}");
    }
  }

  // Read a template document, replace some text, and save as another document
  File.Copy("movie_data.docx", "clone.docx", true);

  int currentRow = 1;
  foreach (Movie movie in movieList)
  {
    ChangeTextInCell("clone.docx", currentRow, 0, movie.Movie_ID);
    ChangeTextInCell("clone.docx", currentRow, 1, movie.Title);
    ChangeTextInCell("clone.docx", currentRow, 2, movie.Release_Date);
    ChangeTextInCell("clone.docx", currentRow, 3, movie.Revenue.ToString("C"));
    ChangeTextInCell("clone.docx", currentRow, 4, movie.Tagline);
    currentRow++;
  }
}
```

```
public static void ChangeTextInCell(string filepath, int rowpos, int cellpos, string text)
{
 using (WordprocessingDocument doc =
    WordprocessingDocument.Open(filepath, true))
  {
    // Find the first table in the document.
    Table table =
      doc.MainDocumentPart.Document.Body.Elements<Table>().First();
    int maxNumberOfColumns = 5;
    //Optional code that determines how many columns are needed in the table
    //foreach (TableRow tr in table.Elements<TableRow>())
    //{
    //  if (tr.Elements<TableCell>().Count() > maxNumberOfColumns)
    //  {
    //    maxNumberOfColumns = tr.Elements<TableCell>().Count();
    //  }
    //}

    // Find the row in the table (add rows and cells if they don't exist).
    if (table.Elements<TableRow>().Count() <= rowpos)
    {
      for (int i = table.Elements<TableRow>().Count() - 1; i < rowpos; i++) {
        var tr = new TableRow();
        table.Append(tr);

      }
    }
    TableRow row =  table.Elements<TableRow>().ElementAt(rowpos);
    for (int j = row.Elements<TableCell>().Count(); j < maxNumberOfColumns; j++)
    {
      var tc = new TableCell();
      row.Append(tc);
    }

    TableCell cell = row.Elements<TableCell>().ElementAt(cellpos);

    // Find the first paragraph in the table cell and add one if necessary.
    if (cell.Elements<Paragraph>().Count() == 0) {
      var para = new Paragraph();
      cell.Append(para);
    }

    Paragraph p = cell.Elements<Paragraph>().First();

    if (p.InnerText == String.Empty)
    {
      string newText = text;
      p.RemoveAllChildren();
      p.AppendChild(new Run(new Text(newText)));
    }
    // Find the first run in the paragraph.
    Run r = p.Elements<Run>().First();

    // Set the text for the run.
    Text t = r.Elements<Text>().First();
    t.Text = text;
  }
}
```

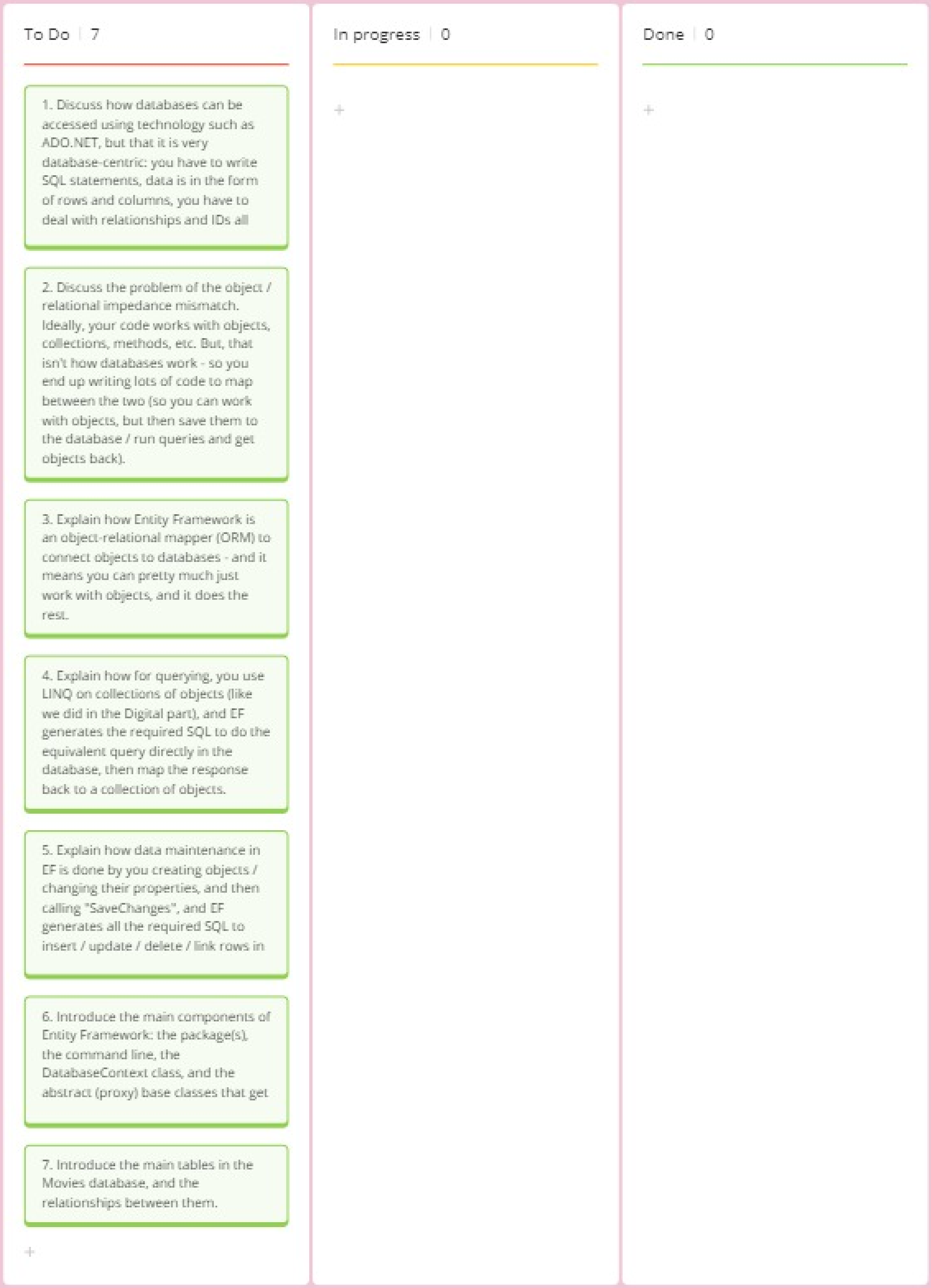


Top revenue earning SciFi movies per alphabetic letter.csv AND movie_data.docx for use with Demo.

Click on documents and select download


Working with databases
Click For Instructions and Resources
Go Back

Introduction to Entity Framework



To Do | 7

1. Discuss how databases can be accessed using technology such as ADO.NET, but that it is very database-centric: you have to write SQL statements, data is in the form of rows and columns, you have to deal with relationships and IDs all

2. Discuss the problem of the object / relational impedance mismatch. Ideally, your code works with objects, collections, methods, etc. But, that isn't how databases work - so you end up writing lots of code to map between the two (so you can work with objects, but then save them to the database / run queries and get objects back).

3. Explain how Entity Framework is an object-relational mapper (ORM) to connect objects to databases - and it means you can pretty much just work with objects, and it does the rest.

4. Explain how for querying, you use LINQ on collections of objects (like we did in the Digital part), and EF generates the required SQL to do the equivalent query directly in the database, then map the response back to a collection of objects.

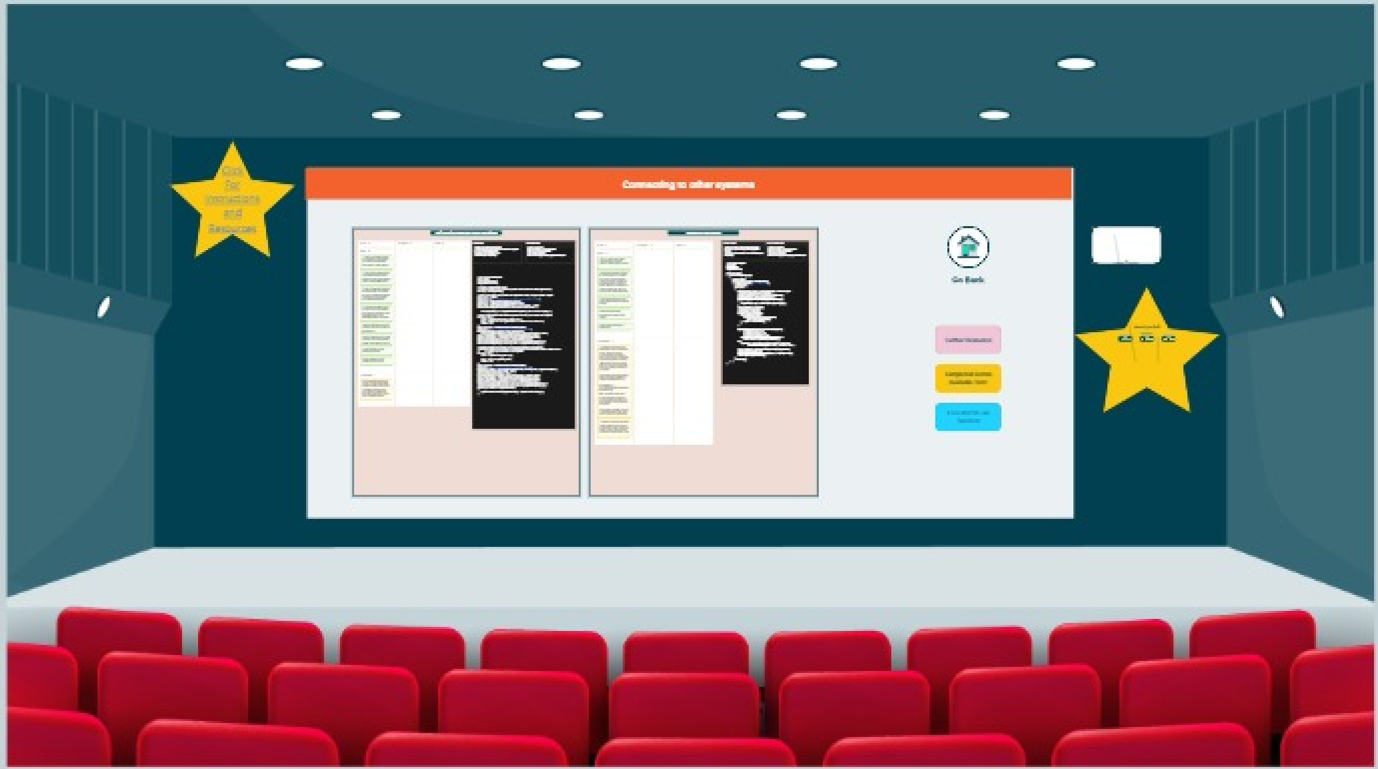
5. Explain how data maintenance in EF is done by you creating objects / changing their properties, and then calling "SaveChanges", and EF generates all the required SQL to insert / update / delete / link rows in

6. Introduce the main components of Entity Framework: the package(s), the command line, the DatabaseContext class, and the abstract (proxy) base classes that get

7. Introduce the main tables in the Movies database, and the relationships between them.

In progress | 0

Done | 0


Click for Instructions and Resources
Connecting to other systems
Go Back

Calling other systems using HttpClient

To Do | 9

Topics | 8

1. Explore some situations where it can be necessary / appropriate for your code to communicate with other systems by making HTTP API

2. Explain how the HttpClient class is a high-level component that will determine the appropriate operating systems provided facilities to best

3. Explain how HttpClient manages a pool of connections, and therefore you should normally only create one instance of it, and re-use it within your application wherever it is

4. Mention that HttpClient provides an asynchronous API, but we'll just be working synchronously for these examples, hence our use of GetResult() to wait for the request's

5. Demonstrate code to send a GET request to a URL, and to process the (JSON) response.

6. Demonstrate code to send a POST request to a URL, including a (JSON) payload, and to collect the response.

7. Show how headers can be attached to the request.

8. Show how headers can be retrieved from the response.

+

Lab activities | 1

The jsonplaceholder.typicode site supports a set of comments (again designed for testing purposes only) that relate to the existing set of posts. See if you can extend the demo code so that it returns a post and its associated comments.

+

In progress | 0

+

+

Done | 0

+

+

Visual Studio

```
//Create a new console project called
//calling-apis-demo.csproj, saving the solution file //project
will live in ConnectingOutSystems

//Use NuGet to add reference to the
//Newtonsoft.Json JSON package
```

Visual Studio Code

```
//Create a new project
dotnet new console -o calling-apis-demo
dotnet sln add calling-apis-demo

//Add required libraries
dotnet add calling-apis-demo package newtonsoft.json
```

```csharp
// Use required namespaces
using System;
using System.Net.Http;
using Newtonsoft.Json;

// Outdoor cinema weather check
// Create an HTTP client (should only have one of these for whole application)
var http = new HttpClient();

// Send a GET request and process the response as JSON, using C# dynamic objects
string city = "Leeds";
string url = $"https://weather-api.coolabs.com/api/weather/{city}";
string json = http.GetStringAsync(url).Result;
dynamic obj = JsonConvert.DeserializeObject(json);
string temp = obj["TemperatureInCelsius"] >= 15 ? "warm" : "cold";
Console.WriteLine($"The weather for the outdoor cinema event " +
    $"in {obj["City"]} is {obj["WeatherDescription"]} and it will be {temp}.");

// Send a POST request with movie blog data to an API, and collect the response
var data = new {
    title = "My movie blog post - Apollo 10.5",
    body = "Apollo 10.5 is a great movie. I'd rate it at 10 out of 10",
    userId = 101
};
string dataJson = JsonConvert.SerializeObject(data);
string url = "https://jsonplaceholder.typicode.com/posts";
HttpResponseMessage response = http.PostAsync(url,
    new StringContent(dataJson, Encoding.UTF8, "application/json")).Result;
string responseJson = response.Content.ReadAsStringAsync().Result;
dynamic responseData = JsonConvert.DeserializeObject(responseJson);
Console.WriteLine($"New post has ID {responseData["id"]}");
Console.WriteLine($"New post has title text of {responseData["title"]}");
Console.WriteLine($"New post has body text of {responseData["body"]}")

// Sending headers with request, and extracting headers from response
// NOTE the typicode site does not do anything with the additional header content and so
//      it does not get returned in the response
var data = new {
    title = "My movie blog post - Up",
    body = "Up is an uplifting film",
    userId = 101
};
string dataJson = JsonConvert.SerializeObject(data);
string url = "https://jsonplaceholder.typicode.com/posts";
HttpContent content = new StringContent(dataJson, Encoding.UTF8, "application/json");
content.Headers.Add("movie_title", "Up");
content.Headers.Add("rating", "10");
HttpResponseMessage response = http.PostAsync(url, content).Result;
string responseJson = response.Content.ReadAsStringAsync().Result;
dynamic responseData = JsonConvert.DeserializeObject(responseJson);
System.Console.WriteLine($"New post has ID {responseData["id"]}");
//Console.WriteLine($"New post has title text of {responseData["title"]}");
//Console.WriteLine($"New post has body text of {responseData["body"]}");
foreach (var header in response.Headers)
{
    System.Console.WriteLine($"{header.Key} = {header.Value.First()}");
}
```

Hosting an API server

To Do | 7

Topics | 5

1. Discuss how APIs work with a server listening for incoming requests, processing them, and then

2. Demonstrate how code can listen for inbound requests on a specific port, and can send a simple text-based response back - explain the concepts of prefixes, starting the listener, waiting for the context, the response object, status code, and output stream. Include loop to keep

3. Demonstrate testing the API by making relevant calls from a tool like Postman.

4. Demonstrate extracting information from the URL of the request.

5. Demonstrate extracting the payload body.

+

Lab activities | 2

1. Implement an API listener that allows clients to book cinema tickets the URL needs to include the number of adults and children, a film_id, date, and time. To begin with, you are encouraged to pass all the relevant data in the URL string (the extension, should you get to it, gets you to pass the data in the body of the request).

Use the HttpListenerContext object's Request.QueryString collection to retrieve the relevant details. e.g.:

ticket.AdultCount = int.Parse(context.Request.QueryString["AdultCount"]);

Return the data in the form of a CustomerTicket that includes the number of adults and children, the film's title, date, time, and total cost of the tickets.

Fake the server side logic such that a hard-coded film title is returned (i.e., there's no need to use a database).

2. Extension: make the API work with a JSON payload and JSON response by passing the number of adults and children, a film_id, date and time in the body of the request. Again, using

+

In progress | 0

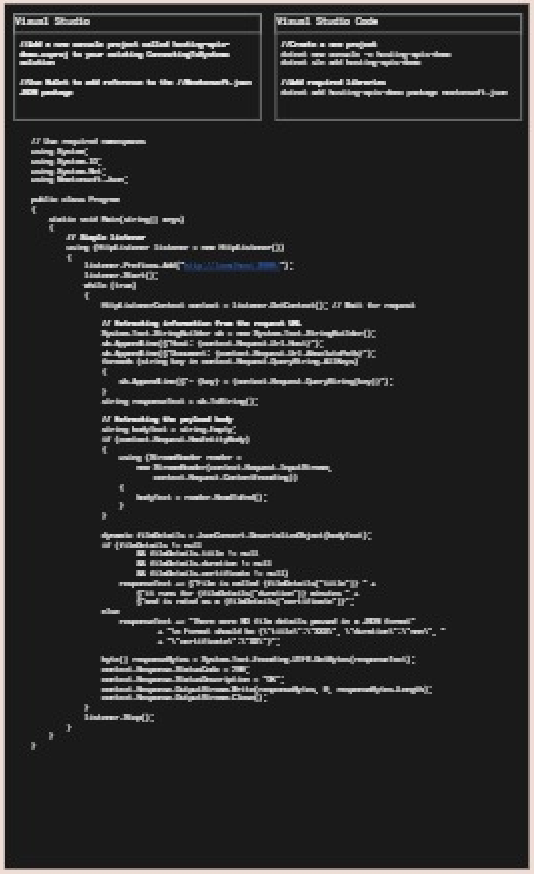
+

+

Done | 0

+

+



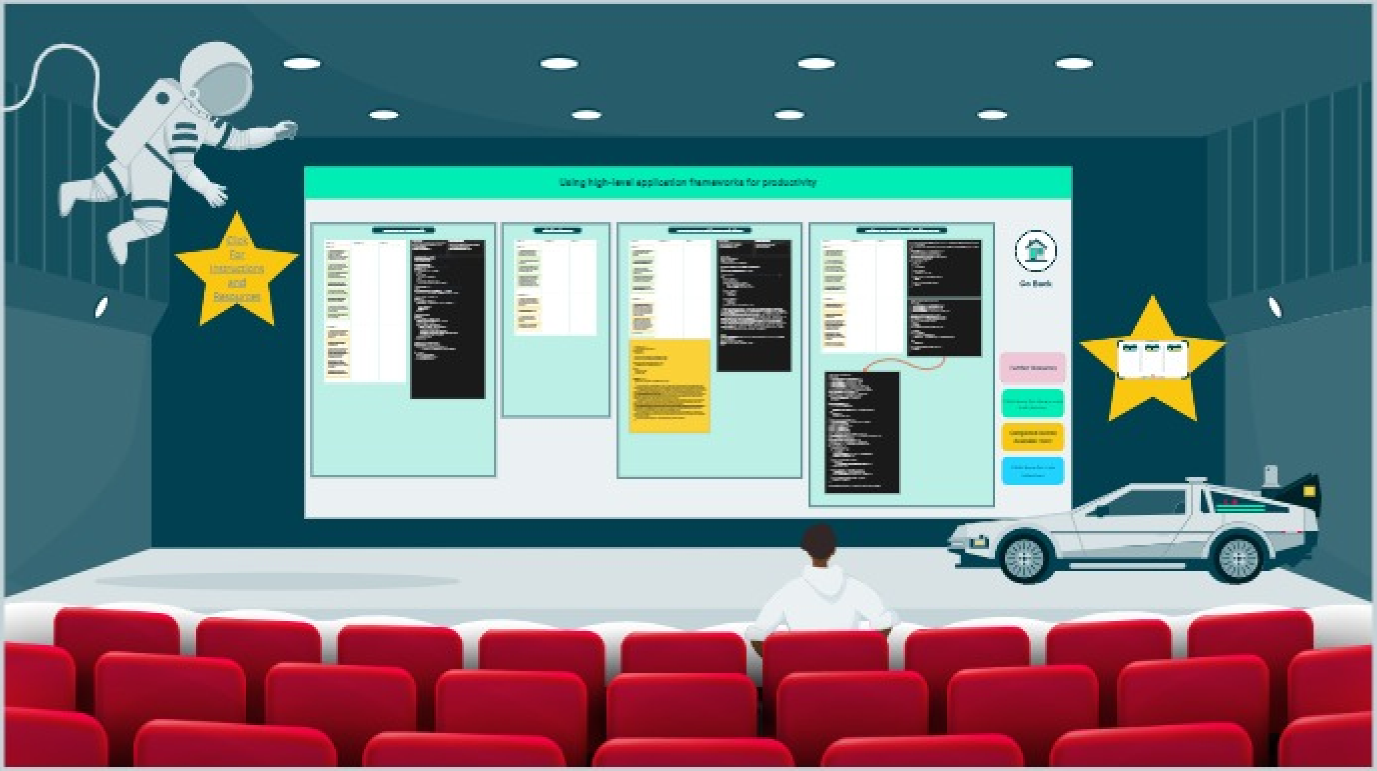
Visual Studio

```
//Add a new console project called hosting-api-demo.csproj to your existing [illegible] solution

//Use NuGet to add reference to the //Newtonsoft.Json JSON package
```

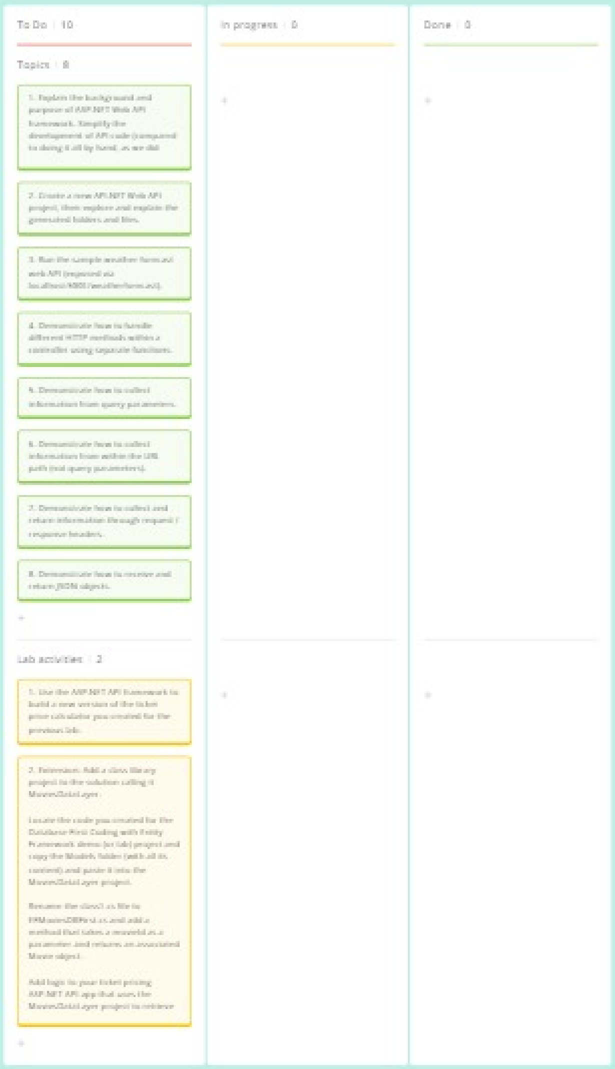
Visual Studio Code

```
//Create a new project
dotnet new console -o hosting-api-demo
dotnet sln add hosting-api-demo

//Add required libraries
dotnet add hosting-api-demo package newtonsoft.json
```

```
// Use required namespaces
using System;
using System.IO;
using System.Net;
using Newtonsoft.Json;

public class Program
{
    static void Main(string[] args)
    {
        // Simple listener
        using (HttpListener listener = new HttpListener())
        {
            listener.Prefixes.Add("http://localhost:8080/");
            listener.Start();
            while (true)
            {
                HttpListenerContext context = listener.GetContext(); // Wait for request

                // Extracting information from the request URL
                System.Text.StringBuilder sb = new System.Text.StringBuilder();
                sb.AppendLine($"Host : {context.Request.Url.Host}");
                sb.AppendLine($"Segment : {context.Request.Url.AbsolutePath}");
                foreach (string key in context.Request.QueryString.AllKeys)
                {
                    sb.AppendLine($" - {key} = {context.Request.QueryString[key]}");
                }
                string responseText = sb.ToString();

                // Extracting the payload body
                string bodyText = string.Empty;
                if (context.Request.HasEntityBody)
                {
                    using (StreamReader reader =
                        new StreamReader(context.Request.InputStream,
                            context.Request.ContentEncoding))
                    {
                        bodyText = reader.ReadToEnd();
                    }
                }

                dynamic filmDetails = JsonConvert.DeserializeObject(bodyText);
                if (filmDetails != null
                        && filmDetails.title != null
                        && filmDetails.duration != null
                        && filmDetails.certificate != null)
                    responseText += $"Film is called {filmDetails["title"]} " +
                        $"It runs for {filmDetails["duration"]} minutes " +
                        $"and is rated as a {filmDetails["certificate"]}";
                else
                    responseText += "There were NO film details passed in a JSON format"
                            + "\n Format should be {\"title\":\"XXXX\", \"duration\":\"nnn\", "
                            + "\"certificate\":\"XX\"}";

                byte[] responseBytes = System.Text.Encoding.UTF8.GetBytes(responseText);
                context.Response.StatusCode = 200;
                context.Response.StatusDescription = "OK";
                context.Response.OutputStream.Write(responseBytes, 0, responseBytes.Length);
                context.Response.OutputStream.Close();
            }
            listener.Stop();
        }
    }
}
```


Using high-level application frameworks for productivity
Click For Instructions and Resources
Go Back
Further Resources

ASP.NET API Framework



To Do | 10

Topics | 8

1. Explain the background and purpose of ASP.NET Web API framework. Simplify the development of API code (compared to doing it all by hand, as we did

2. Create a new API.NET Web API project, then explore and explain the generated folders and files.

3. Run the sample weather forecast web API (exposed via localhost:5000/weatherforecast).

4. Demonstrate how to handle different HTTP methods within a controller using separate functions.

5. Demonstrate how to collect information from query parameters.

6. Demonstrate how to collect information from within the URL path (not query parameters).

7. Demonstrate how to collect and return information through request / response headers.

8. Demonstrate how to receive and return JSON objects.

+

Lab activities | 2

1. Use the ASP.NET API framework to build a new version of the ticket price calculator you created for the previous lab.

2. Extension: Add a class library project to the solution calling it MoviesDataLayer.

Locate the code you created for the Database First Coding with Entity Framework demo (or lab) project and copy the Models folder (with all its content) and paste it into the MoviesDataLayer project.

Rename the class1.cs file to EFMoviesDBFirst.cs and add a method that takes a movieId as a parameter and returns an associated Movie object.

Add logic to your ticket pricing ASP.NET API app that uses the MoviesDataLayer project to retrieve

+

In progress | 0

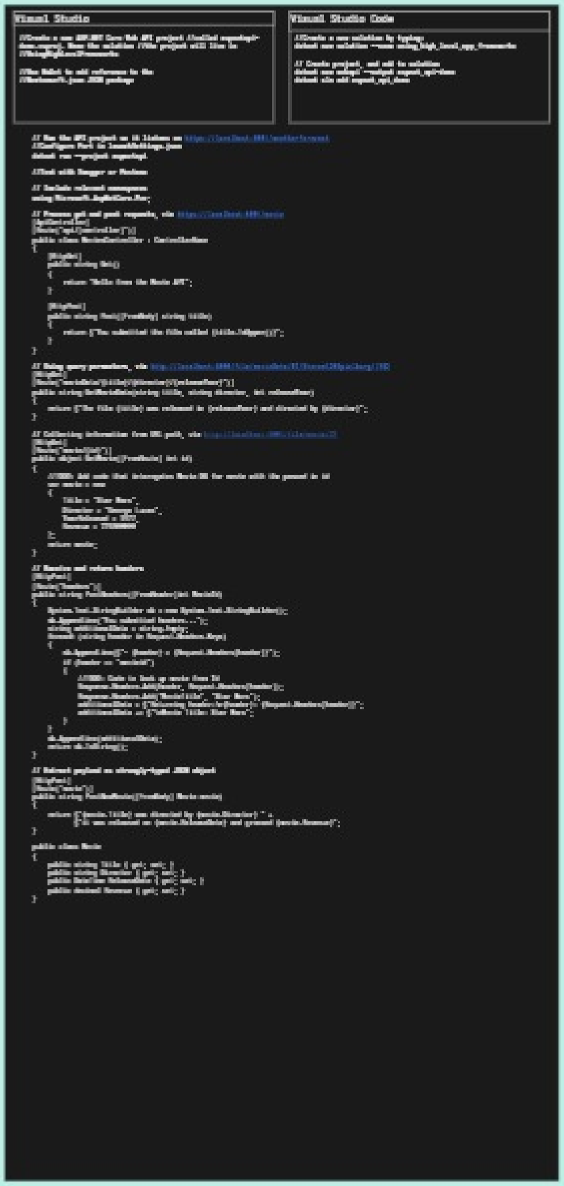
+

+

Done | 0

+

+



Visual Studio

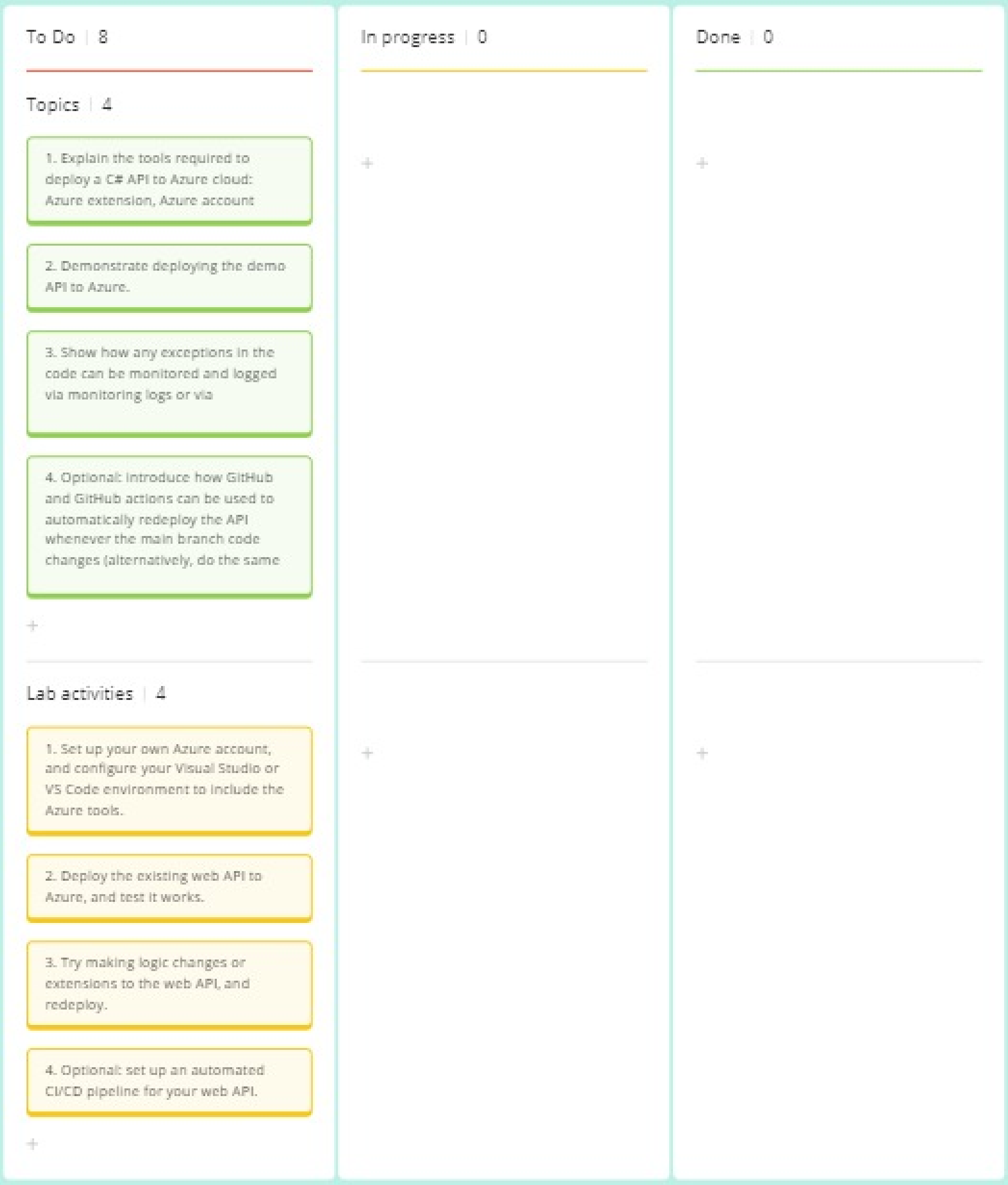
```
//Create a new ASP.NET Core Web API project (called [illegible]
[illegible]). Name the solution //the project will live in
//[illegible]

//Use NuGet to add reference to the
//Microsoft.[illegible] package
```

Visual Studio Code

```
//Create a new solution by typing:
dotnet new solution --name [illegible]

// Create project, and add to solution
dotnet new webapi --output [illegible]
dotnet sln add [illegible]
```

```
// Run the API project so it listens on [illegible]/weatherforecast
//Configure Port in launchSettings.json
dotnet run --project [illegible]

//Test with Swagger or Postman

// Include relevant namespace
using Microsoft.AspNetCore.Mvc;

// Process get and post requests, via [illegible]/movie
[ApiController]
[Route("api/[controller]")]
public class MoviesController : ControllerBase
{
    [HttpGet]
    public string Get()
    {
        return "Hello from the Movie API";
    }

    [HttpPost]
    public string Post([FromBody] string title)
    {
        return $"You submitted the title called {title.ToUpper()}";
    }
}

// Using query parameters, via [illegible]
[HttpGet]
[Route("movieInfo/{title}/{director}/{releaseYear}")]
public string GetMovieInfo(string title, string director, int releaseYear)
{
    return $"The film {title} was released in {releaseYear} and directed by {director}";
}

// Collecting information from URL path, via [illegible]
[HttpGet]
[Route("movie/{id}")]
public object GetMovie([FromRoute] int id)
{
    //TODO: Add code that interrogates MovieDB for movie with the passed in id
    var movie = new
    {
        Title = "Star Wars",
        Director = "George Lucas",
        YearReleased = [illegible],
        Revenue = [illegible]
    };
    return movie;
}

// Receive and return headers
[HttpPost]
[Route("headers")]
public string PostHeaders([FromHeader]int MovieId)
{
    System.Text.StringBuilder sb = new System.Text.StringBuilder();
    sb.AppendLine("You submitted headers...");
    string additionalData = string.Empty;
    foreach (string header in Request.Headers.Keys)
    {
        sb.AppendLine($"- {header} = {Request.Headers[header]}");
        if (header == "movieId")
        {
            //TODO: Code to look up movie from DB
            Response.Headers.Add(header, Request.Headers[header]);
            Response.Headers.Add("MovieTitle", "Star Wars");
            additionalData = $"Returning header:\n{header}: {Request.Headers[header]}";
            additionalData += $"\nMovie Title: Star Wars";
        }
    }
    sb.AppendLine(additionalData);
    return sb.ToString();
}

// Entered payload as strongly-typed JSON object
[HttpPost]
[Route("movie")]
public string PostNewMovie([FromBody] Movie movie)
{
    return $"{movie.Title} was directed by {movie.Director}" +
           $"It was released on {movie.ReleaseDate} and grossed {movie.Revenue}";
}

public class Movie
{
    public string Title { get; set; }
    public string Director { get; set; }
    public DateTime ReleaseDate { get; set; }
    public decimal Revenue { get; set; }
}
```

Cloud Deployment



To Do | 8

Topics | 4

1. Explain the tools required to deploy a C# API to Azure cloud: Azure extension, Azure account

2. Demonstrate deploying the demo API to Azure.

3. Show how any exceptions in the code can be monitored and logged via monitoring logs or via

4. Optional: introduce how GitHub and GitHub actions can be used to automatically redeploy the API whenever the main branch code changes (alternatively, do the same

Lab activities | 4

1. Set up your own Azure account, and configure your Visual Studio or VS Code environment to include the Azure tools.

2. Deploy the existing web API to Azure, and test it works.

3. Try making logic changes or extensions to the web API, and redeploy.

4. Optional: set up an automated CI/CD pipeline for your web API.

In progress | 0

Done | 0

Running C# in a web browser with Blazor

To Do | 8

In progress | 0

Done | 0

Topics | 5

1. Create a new Blazor project, and explore the folders and files generated.

2. Run the default Blazor application, and see how it behaves (especially the counter button feature).

3. Highlight the code within Counter.razor that implements the behaviour seen.

4. Demonstrate how UI layout could be changed to include an increasing number of stars, based on clicks.

5. Show how input boxes can be wired to variables, so C# code can be used to do calculations.

Lab activities | 3

1. Using the code snippet below as a start, implement a birth year calculator UI. The user should input their name and age and then click a "year born" button which calculates the users birth year and displays it beneath the button.

2. Implement a more complex calculator, that presents the user with the name of a film that was released in the year of their birth by making use of the films dictionary used by the film rater app.

Type something

```
<!-- Code for part 1 -->
<!--  Using data from input boxes -->
@page "/inputs"

<h1>Inputs</h1>

<input type="text" placeholder="Name" @bind="name">
<input type="number" placeholder="Age" @bind="age">

@if (!string.IsNullOrEmpty(name) && age != null) {
 <p>Hello @name, you are @age years old!</p>
}

@code {
   private string name;
   private int? age;
}

<!-- Code for part 2 -->
@code {
   Dictionary<int, string> films = new Dictionary<int, string>
   {
       {1960, "Psycho"},{1961, "West Side Story"},{1962, "Dr. No"},{1963, "The Great Escape"},{1964, "Goldfinger"},{1965,
"Thunderball"},{1966, "Batman"},{1967, "You Only Live Twice"},{1968, "Oliver!"},{1969, "On Her Majesty's Secret Service"},
       {1970, "Beneath the Planet of the Apes"},{1971, "Diamonds Are Forever"},{1972, "The Godfather"},{1973, "Live and Let
Die"},{1974, "The Man with the Golden Gun"},{1975, "Monty Python and the Holy Grail"},{1976, "Rocky"},{1977, "Star
Wars"},{1978, "Grease"},{1979, "Moonraker"},
       {1980, "The Blues Brothers"},{1981, "Raiders of the Lost Ark"},{1982, "E.T. the Extra-Terrestrial"},{1983, "Return of the
Jedi"},{1984, "Indiana Jones and the Temple of Doom"},{1985, "A View to a Kill"},{1986, "Top Gun"},{1987, "The Living
Daylights"},{1988, "Die Hard"},{1989, "Indiana Jones and the Last Crusade"},
       {1990, "Die Hard 2"},{1991, "Robin Hood: Prince of Thieves"},{1992, "Wayne's World"},{1993, "Jurassic Park"},{1994,
"Four Weddings and a Funeral"},{1995, "Die Hard: With a Vengeance"},{1996, "Mission: Impossible"},{1997, "Men in
Black"},{1998, "Armageddon"},{1999, "Star Wars: Episode I - The Phantom Menace"},
       {2000, "Gladiator"},{2001, "Harry Potter and the Philosopher's Stone"},{2002, "Harry Potter and the Chamber of
Secrets"},{2003, "Finding Nemo"},{2004, "Harry Potter and the Prisoner of Azkaban"},{2005, "Harry Potter and the Goblet
of Fire"},{2006, "Casino Royale"},{2007, "Harry Potter and the Order of the Phoenix"},{2008, "Indiana Jones and the
Kingdom of the Crystal Skull"},{2009, "Harry Potter and the Half-Blood Prince"},
       {2010, "Inception"},{2011, "Pirates of the Caribbean: On Stranger Tides"},{2012, "Skyfall"},{2013, "Frozen"},{2014,
"Guardians of the Galaxy"},{2015, "Minions"},{2016, "Finding Dory"},{2017, "Star Wars: The Last Jedi"},{2018, "Incredibles
2"},{2019, "The Lion King"},
       {2020, "The Eight Hundred"},{2021, "No Time to Die"},{2022, "Top Gun: Maverick"},
   };
}
```

Visual Studio

```
//Add a new Blazor Web Assembly App project
//called blazor_demo.csproj to the existing
//UsingHighLevelFrameworks solution
```

Visual Studio Code

```
//Start a new Blazor WASM project and add to solution
dotnet new blazorwasm -o blazor_demo
dotnet sln add blazor_demo
```

```
// Create a Film Rater page using Blazor
@page "/filmRater"

<PageTitle>Film Rater</PageTitle>

<h1>Rate: @FilmTitle</h1>

<p role="status">Current rating: @currentRating</p>

<button class="btn btn-primary" @onclick="IncrementCount">Rate Up</button>
<button class="btn btn-primary" @onclick="DecrementCount">Rate Down</button>
<br />

<!-- Show a number of stars based on the number of clicks -->
@for (int i = 0; i < currentRating; i++)
{
   <img width="32px" src="[illegible]">
}

@code {
   private int currentRating = 0;

   string FilmTitle { get; set; }

   protected override async Task OnInitializedAsync()
   {
          Random r = new Random(DateTime.Now.Millisecond);
          int year = r.Next(1960, 1960 + films.Count);
          FilmTitle = films[year];
   }

   private void IncrementCount()
   {
      if(currentRating < 10)
          currentRating++;
   }


   private void DecrementCount()
   {
      if(currentRating > 0)
          currentRating--;
   }

   Dictionary<int, string> films = new Dictionary<int, string>
   {
      {1960, "Psycho"},{1961, "West Side Story"},{1962, "Dr. No"},{1963, "The Great Escape"},{1964, "Goldfinger"},{1965,
 "Thunderball"},{1966, "Batman"},{1967, "You Only Live Twice"},{1968, "Oliver!"},{1969, "On Her Majesty's Secret Service"},
      {1970, "Beneath the Planet of the Apes"},{1971, "Diamonds Are Forever"},{1972, "The Godfather"},{1973, "Live and Let
Die"},{1974, "The Man with the Golden Gun"},{1975, "Monty Python and the Holy Grail"},{1976, "Rocky"},{1977, "Star Wars"},{1978,
 "Grease"},{1979, "Moonraker"},
      {1980, "The Blues Brothers"},{1981, "Raiders of the Lost Ark"},{1982, "E.T. the Extra-Terrestrial"},{1983, "Return of the
Jedi"},{1984, "Indiana Jones and the Temple of Doom"},{1985, "A View to a Kill"},{1986, "Top Gun"},{1987, "The Living
Daylights"},{1988, "Die Hard"},{1989, "Indiana Jones and the Last Crusade"},
      {1990, "Die Hard 2"},{1991, "Robin Hood: Prince of Thieves"},{1992, "Wayne's World"},{1993, "Jurassic Park"},{1994, "Four
Weddings and a Funeral"},{1995, "Die Hard: With a Vengeance"},{1996, "Mission: Impossible"},{1997, "Men in Black"},{1998,
 "Armageddon"},{1999, "Star Wars: Episode I - The Phantom Menace"},
      {2000, "Gladiator"},{2001, "Harry Potter and the Philosopher's Stone"},{2002, "Harry Potter and the Chamber of
Secrets"},{2003, "Finding Nemo"},{2004, "Harry Potter and the Prisoner of Azkaban"},{2005, "Harry Potter and the Goblet of
Fire"},{2006, "Casino Royale"},{2007, "Harry Potter and the Order of the Phoenix"},{2008, "Indiana Jones and the Kingdom of the
Crystal Skull"},{2009, "Harry Potter and the Half-Blood Prince"},
      {2010, "Inception"},{2011, "Pirates of the Caribbean: On Stranger Tides"},{2012, "Skyfall"},{2013, "Frozen"},{2014,
 "Guardians of the Galaxy"},{2015, "Minions"},{2016, "Finding Dory"},{2017, "Star Wars: The Last Jedi"},{2018, "Incredibles
2"},{2019, "The Lion King"},
      {2020, "The Eight Hundred"},{2021, "No Time to Die"},{2022, "Top Gun: Maverick"},
   };
}



Adding NAV
//Open the file called NavMenu.razor located in the project's Shared folder and add the following code within the nav element
alongside the other NavLinks and divs:

<div class="nav-item px-3">
<NavLink class="nav-link" href="filmRater">
<span class="oi oi-plus" aria-hidden="true"></span> Film Rater
</NavLink>
</div>
```

Desktop GUI app with C# and Windows Forms

To Do | 7

Topics | 5

1. Show how a C# project can be created in Visual Studio (not VS Code) for a Windows Forms app.

2. Explore the default folders and files, and run the initial application.

3. Show how code be added to events on controls, such as button clicks, and how properties of controls.

4. Show how one form can create another form, for example when drilling down to more details.

5. Show how data can be retrieved from input controls and used as part of C# logic and calculations.

+

Lab activities | 2

1. Build an Windows Forms application to create a new version of the ticket price calculator you

2. Extension: Add a class library project to the solution calling it MoviesDataLayer.

Locate the code you created for the Database First Coding with Entity Framework demo (or lab) project and copy the Models folder (with all its content) and paste it into the MoviesDataLayer project.

Rename the class1.cs file to EFMoviesDBFirst.cs and add a method that takes a movieId as a parameter and returns an associated Movie object.

Add logic to your ticket pricing API app that uses the MoviesDataLayer

+

In progress | 0

+

+

Done | 0

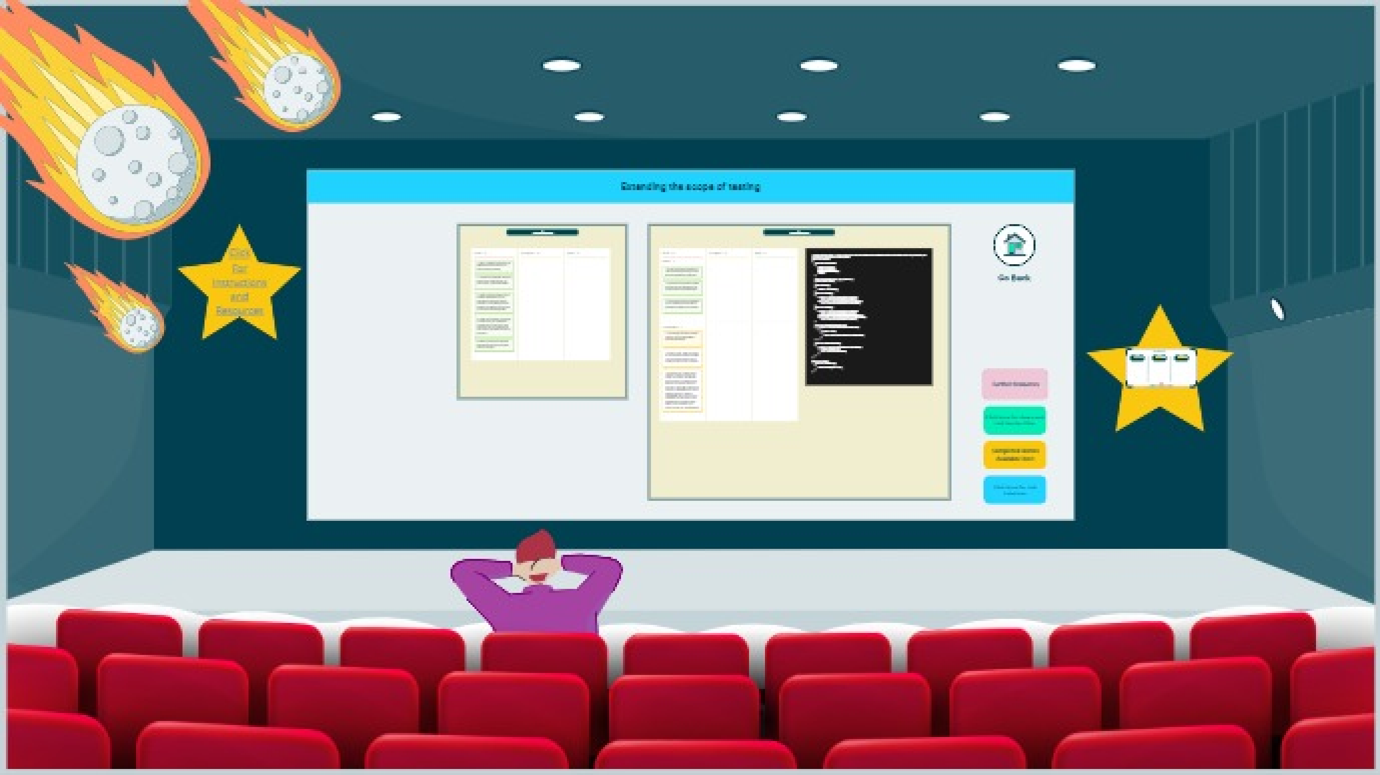
+

+

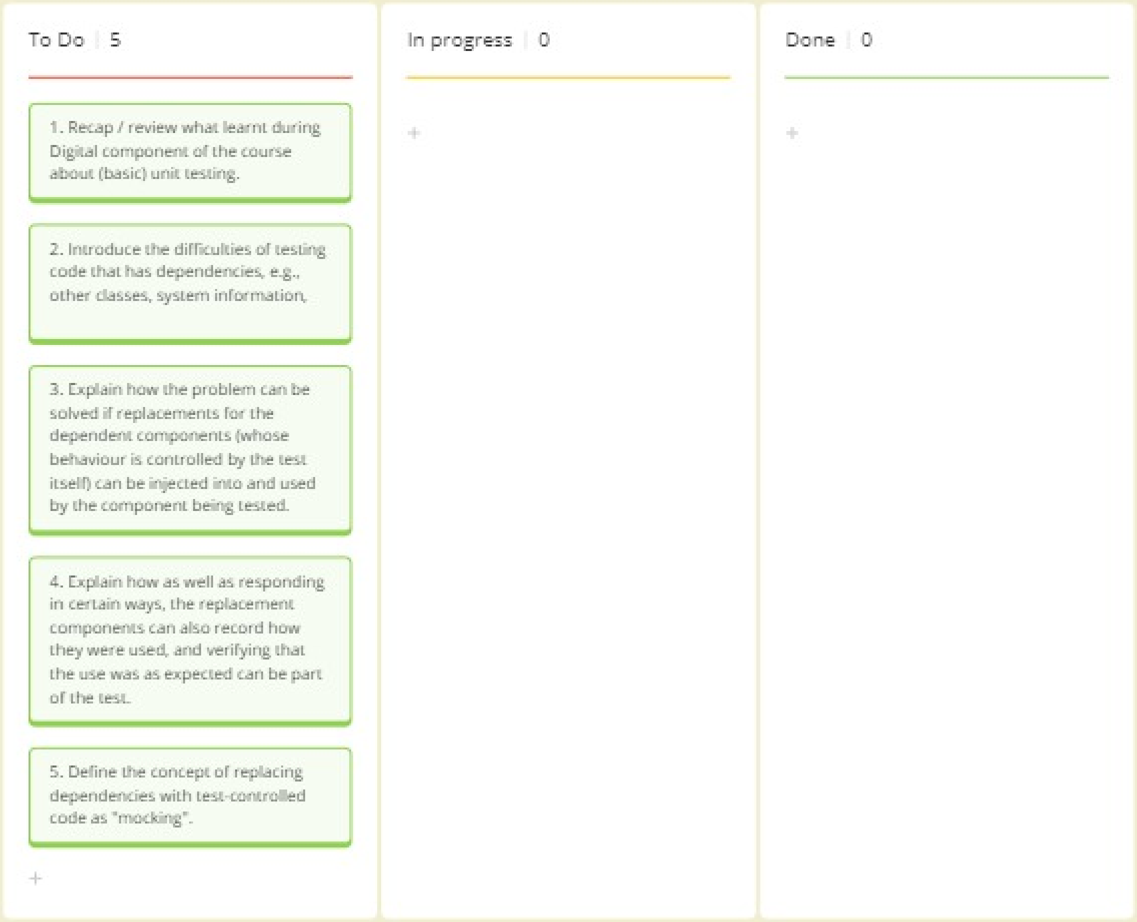
```
NOTE: Windows Forms apps can ONLY be created using Visual Studio (not VSC). They use the .NET Framework (rather than .NET 6) so so can
only run on Windows machines.

//Start by using Visual Studio to add a new Windows Forms App project to the existing UsingHighLevelFrameworks solution (if you have
one)

//Ensure Form has:
//    Button called CountButton with its Text property set to "OK"
//    Label called CountLabel With its Text property set to 0
//
// Add the following lines of code to the Form1.cs file and
// use the Design and Property windows to hook
// the CountButton object up to the CountButton_Click event handler and
// Form1 up to the Form1_Load event handler.

public partial class Form1 : Form
{
    private int count = 0;

    public Form1()
    {
        InitializeComponent();
    }

    private void UpdateCount()
    {
        CountLabel.Text = $"Count: {count}";
    }

    private void CountButton_Click(object sender, EventArgs e)
    {
        count++;
        UpdateCount();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        UpdateCount();
    }
}
```

```
// Example of retrieving information from input controls

//Ensure Form has:
//    Button called CountButton with its Text property set to "OK"
//    Label called CountLabel with its Text property set to 0
//    Label called NameLabel with its Text property set to "Name: "
//    TextBox called NameTextBox
//    Label called AgeLabel with its Text property set to "Age: "
//    NumericUpDown called AgeUpDown
//    Label called MessageLabel with its Text property set to an empty string
//

// Add the following lines of code to the Form1.cs file and
// use the Design and Property windows to hook the
// NameTextBox control up to the NameTextBox_TextChanged event handler
// and the AgeUpDown control up to the AgeUpDown_ValueChanged event handler.

private void Form1_Load(object sender, EventArgs e)
{
    UpdateCount();
    UpdateMessage();
}

private void NameTextBox_TextChanged(object sender, EventArgs e)
{
    UpdateMessage();
}

private void UpdateMessage()
{
    if (!string.IsNullOrEmpty(NameTextBox.Text))
    {
        MessageLabel.Text = $"Hello {NameTextBox.Text}, you are {AgeUpDown.Text} years old!";
    }
    else
    {
        MessageLabel.Text = "";
    }
}

private void AgeUpDown_ValueChanged(object sender, EventArgs e)
{
    UpdateMessage();
}
```

```
// Example of one form loading another one

//Ensure Form1 has:
//    Button called CountButton with its Text property set to "OK"
//    Label called CountLabel with its Text property set to 0
//    Label called NameLabel with its Text property set to "Name: "
//    TextBox called NameTextBox
//    Label called AgeLabel with its Text property set to "Age: "
//    NumericUpDown called AgeUpDown
//    Label called MessageLabel with its Text property set to an empty string
//    Button called ShowMoviesButton with its Text property set to "Show Movies"

// Add the following lines of code to the Form1.cs file (making sure to replace the
// existing UpdateMessage function with new version). Then
// use the Design and Property windows to hook the
// ShowMoviesButton control up to the ShowMoviesButton_Click event handler

private void ShowMoviesButton_Click(object sender, EventArgs e)
{
    int YearOfBirth = DateTime.Now.Year - (int)(AgeUpDown.Value);
    Form2 form = new Form2(YearOfBirth);
    form.ShowDialog();
}

// Replace existing UpdateMessage code
private void UpdateMessage()
{
    if (!string.IsNullOrEmpty(NameTextBox.Text))
    {
        MessageLabel.Text = $"Hello {NameTextBox.Text}, you are {AgeUpDown.Value} years old!";
        ShowMoviesButton.Enabled = true;
    }
    else
    {
        MessageLabel.Text = "";
        ShowMoviesButton.Enabled = false;
    }
}

// Add a new Form2 to the project and ensure it has:
//    Button called LoadMoviesButton with its Text property set to "Load Movies"
//    DataGridView called MoviesDGV
//    NumericUpDown called DobMovieYear

// Use NuGet to add references to the following:
// Microsoft.EntityFrameworkCore
// Microsoft.EntityFrameworkCore.Design
// Microsoft.EntityFrameworkCore.SqlServer
// Microsoft.EntityFrameworkCore.Tools
// Microsoft.Extensions.Configuration.Json

// NOTE: For Model classes to be automatically generated run the following
// from the Package Manager Console:
// dotnet-ef dbcontext scaffold "Data Source=(local); Initial Catalog=Movies; trusted_connection=true"
Microsoft.EntityFrameworkCore.SqlServer -o Models

// Add the following lines of code to the Form2.cs file. Then
// use the Design and Property windows to hook the
// LoadMoviesButton control up to the LoadMoviesButton_Click event handler
// and the DobMovieYear control to the DobMovieYear_ValueChanged event handler

public partial class Form2 : Form
{
    private MoviesContext context = null;

    public Form2(int yearOfBirth = 1900)
    {
        InitializeComponent();
        context = new MoviesContext();
        if (yearOfBirth < DobMovieYear.Minimum || yearOfBirth > DobMovieYear.Maximum)
            yearOfBirth = (int)DobMovieYear.Minimum;
        DobMovieYear.Value = yearOfBirth;
    }

    private void Form2_Load(object sender, EventArgs e)
    {
        List<Movie> movies =
            context.Movies.Where(m => ((DateTime)m.ReleaseDate).Year
                == (int)(DobMovieYear.Value)).OrderByDescending(m => m.Revenue).ToList();
        MoviesDGV.DataSource = movies;
    }

    private void LoadMoviesButton_Click(object sender, EventArgs e)
    {
        List<Movie> movies =
            context.Movies.Where(m => ((DateTime)m.ReleaseDate).Year
                == (int)(DobMovieYear.Value)).OrderByDescending(m => m.Revenue).ToList();
        MoviesDGV.DataSource = movies;
    }

    private void DobMovieYear_ValueChanged(object sender, EventArgs e)
    {
        LoadMoviesButton_Click(sender, e);
    }
}

//We Found there are quite a few years for which there are no corresponding movies in the database.
```


Expanding the scope of testing
Click For Instructions and Resources
Go Back

Introduction



To Do | 5

1. Recap / review what learnt during Digital component of the course about (basic) unit testing.

2. Introduce the difficulties of testing code that has dependencies, e.g., other classes, system information,

3. Explain how the problem can be solved if replacements for the dependent components (whose behaviour is controlled by the test itself) can be injected into and used by the component being tested.

4. Explain how as well as responding in certain ways, the replacement components can also record how they were used, and verifying that the use was as expected can be part of the test.

5. Define the concept of replacing dependencies with test-controlled code as "mocking".

In progress | 0

Done | 0

Introduction

To Do | 6

Topics | 3

1. Introduce the code sample to be used for exploring mocking: missile launcher (download from GitHub).

2. Explore how manual mock objects could be built and injected into the missile system to enable testing of

3. Explore how a mocking framework such as FakeItEasy can be used to simplify the creation of the mocked

Lab activities | 3

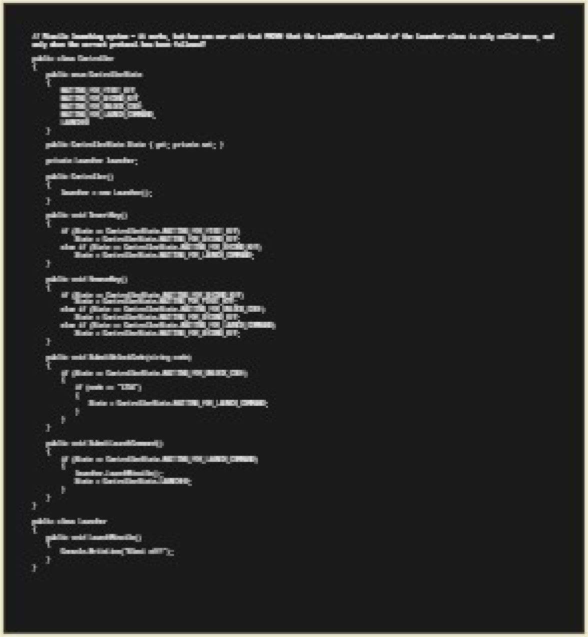
1. Use mocking techniques to write unit tests for the Movie Selector Visual Studio Solution.

2. The link in step 1 gives you access to a ready written project that allows users to see what movies that are showing based on the current time

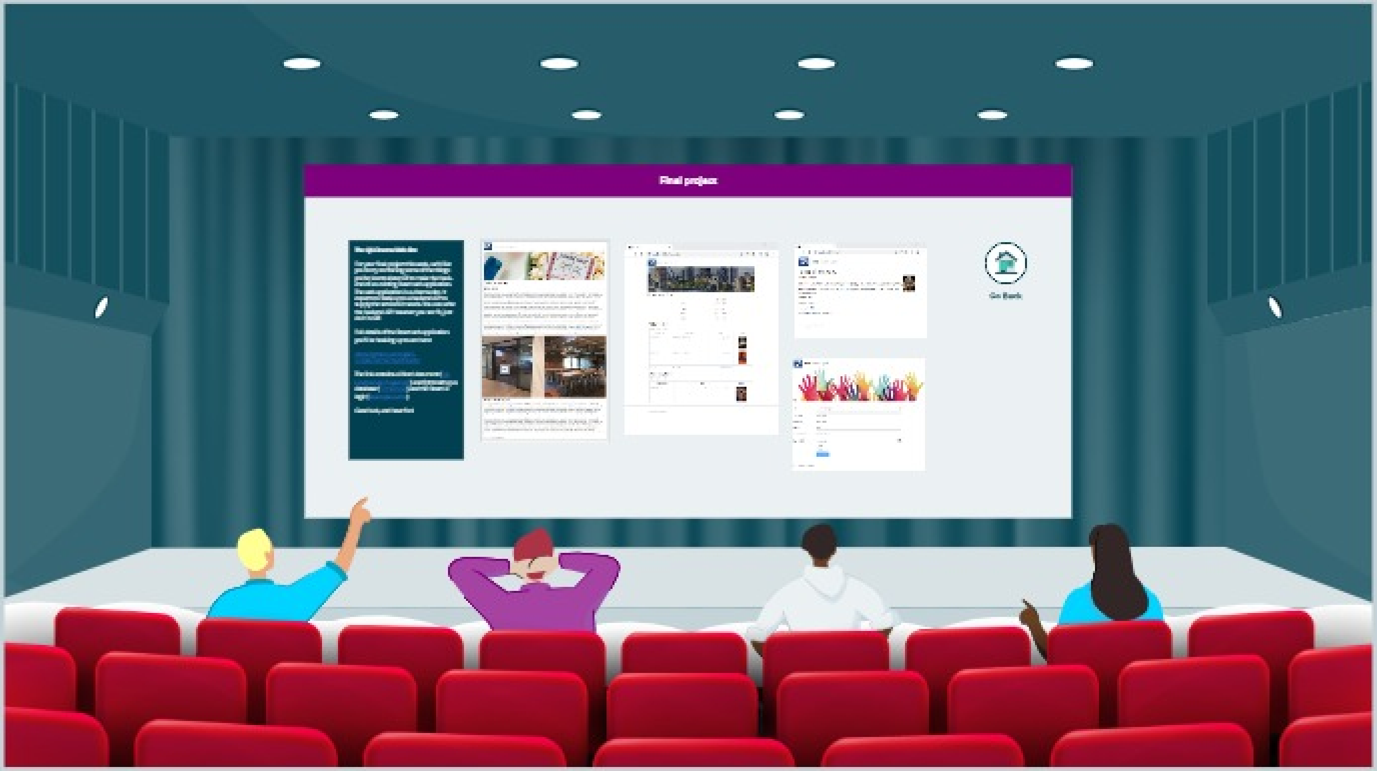
3. All need to do is create a set of unit tests that ensure the correct movies are served up at different times of day and on different days (based on the number of days on from the current date). A look at the Program.cs file will give you an idea of What is going on. Note the MovieDecider class's constructor is overloaded and one version takes ITimeService and IMoviesContext objects as its parameters. This means the class has 2 dependencies.

In progress | 0

Done | 0



```
// Missile launching system - it works, but how can we unit test WITH that the LaunchMissile method of the Launcher class is only called once, and
only when the correct protocol has been followed?
public class Controller
{
    public enum ControllerState
    {
        WAITING_FOR_FIRST_KEY,
        WAITING_FOR_SECOND_KEY,
        WAITING_FOR_UNLOCK_CODE,
        WAITING_FOR_LAUNCH_COMMAND,
        LAUNCHED
    }

    public ControllerState State { get; private set; }

    private Launcher launcher;

    public Controller()
    {
        launcher = new Launcher();
    }

    public void TurnKeyOn()
    {
        if (State == ControllerState.WAITING_FOR_FIRST_KEY)
            State = ControllerState.WAITING_FOR_SECOND_KEY;
        else if (State == ControllerState.WAITING_FOR_SECOND_KEY)
            State = ControllerState.WAITING_FOR_UNLOCK_CODE;
    }

    public void TurnKeyOff()
    {
        if (State == ControllerState.WAITING_FOR_SECOND_KEY)
            State = ControllerState.WAITING_FOR_FIRST_KEY;
        else if (State == ControllerState.WAITING_FOR_UNLOCK_CODE)
            State = ControllerState.WAITING_FOR_FIRST_KEY;
        else if (State == ControllerState.WAITING_FOR_LAUNCH_COMMAND)
            State = ControllerState.WAITING_FOR_FIRST_KEY;
    }

    public void SubmitUnlockCode(string code)
    {
        if (State == ControllerState.WAITING_FOR_UNLOCK_CODE)
        {
            if (code == "1234")
            {
                State = ControllerState.WAITING_FOR_LAUNCH_COMMAND;
            }
        }
    }

    public void SubmitLaunchCommand()
    {
        if (State == ControllerState.WAITING_FOR_LAUNCH_COMMAND)
        {
            launcher.LaunchMissile();
            State = ControllerState.LAUNCHED;
        }
    }
}

public class Launcher
{
    public void LaunchMissile()
    {
        Console.WriteLine("Blast off!");
    }
}
```


Final project
Go Back

The QA Cinema Web Site

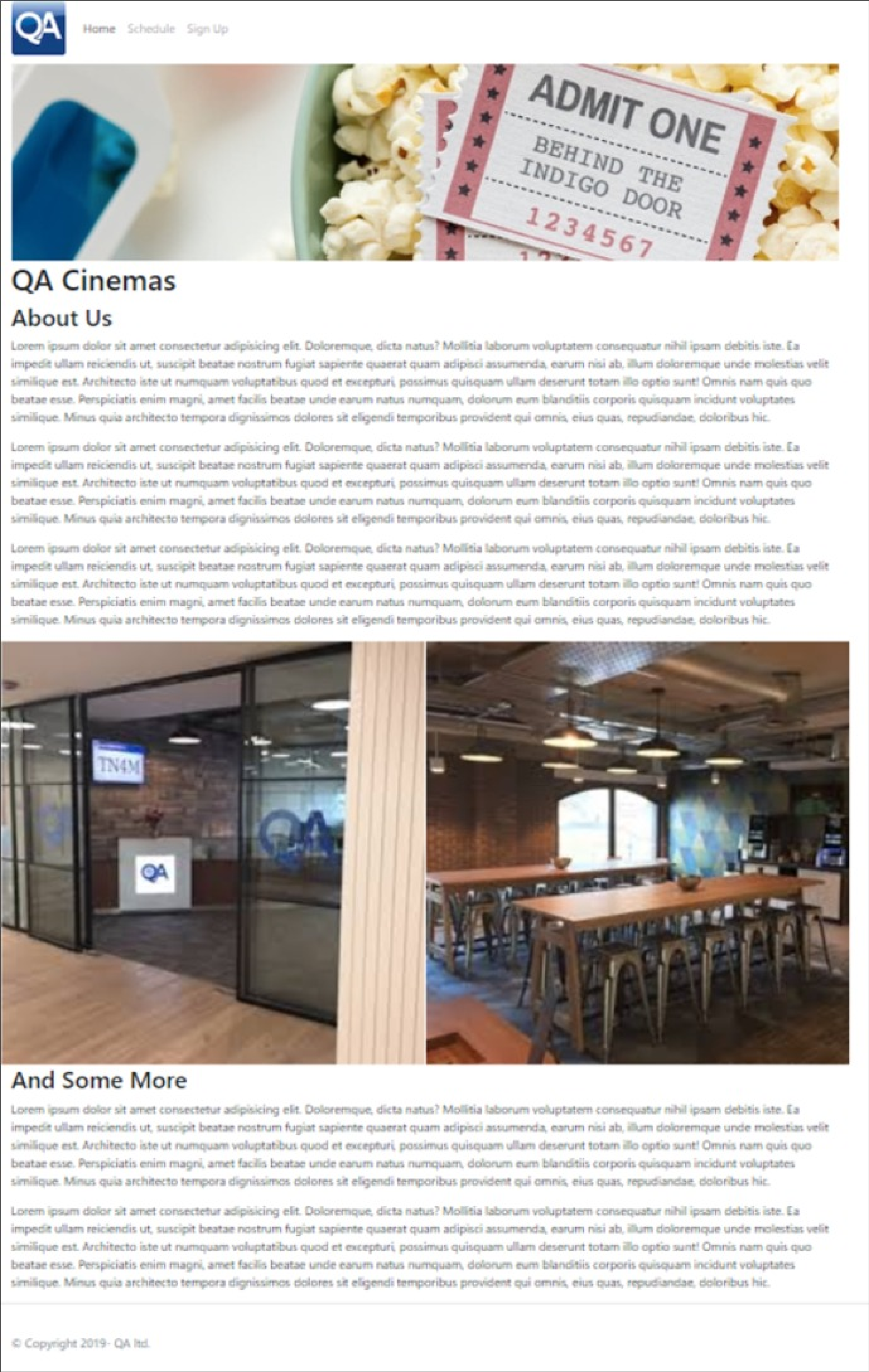
For your final project this week, we'd like you to try combining some of the things you've learnt about C# to make the back-end of an existing React web application. The web application is a cinema site. It expects to hook up to a backend API to supply the services it needs. You can write the backend API however you see fit, just do it in C#!

Full details of the React web application you'll be hooking up to are here:

https://github.com/QACS-TL/QACinemaProjectStarter

The link contains a Word document (QA Cinema Site Project.pdf) a script to set up a database (FilmsDB.sql) and the React UI logic (ReactQACinema)

Good luck, and have fun!



QA Cinemas

About Us

Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque, dicta natus? Mollitia laborum voluptatem consequatur nihil ipsam debitis iste. Ea impedit ullam reiciendis ut, suscipit beatae nostrum fugiat sapiente quaerat quam adipisci assumenda, earum nisi ab, illum doloremque unde molestias velit similique est. Architecto iste ut numquam voluptatibus quod et excepturi, possimus quisquam ullam deserunt totam illo optio sunt! Omnis nam quis quo beatae esse. Perspiciatis enim magni, amet facilis beatae unde earum natus numquam, dolorum eum blanditiis corporis quisquam incidunt voluptates similique. Minus quia architecto tempora dignissimos dolores sit eligendi temporibus provident qui omnis, eius quas, repudiandae, doloribus hic.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque, dicta natus? Mollitia laborum voluptatem consequatur nihil ipsam debitis iste. Ea impedit ullam reiciendis ut, suscipit beatae nostrum fugiat sapiente quaerat quam adipisci assumenda, earum nisi ab, illum doloremque unde molestias velit similique est. Architecto iste ut numquam voluptatibus quod et excepturi, possimus quisquam ullam deserunt totam illo optio sunt! Omnis nam quis quo beatae esse. Perspiciatis enim magni, amet facilis beatae unde earum natus numquam, dolorum eum blanditiis corporis quisquam incidunt voluptates similique. Minus quia architecto tempora dignissimos dolores sit eligendi temporibus provident qui omnis, eius quas, repudiandae, doloribus hic.

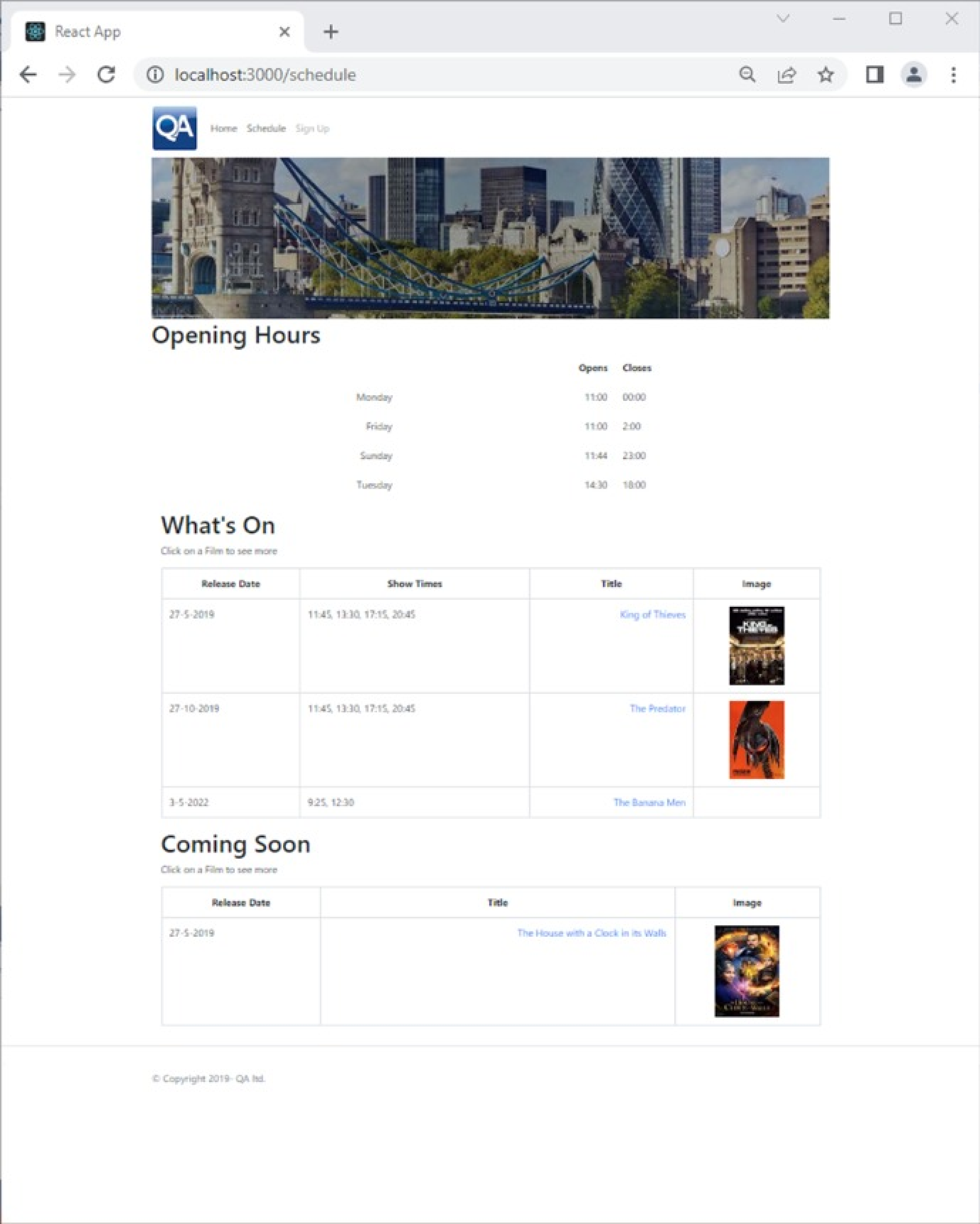
Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque, dicta natus? Mollitia laborum voluptatem consequatur nihil ipsam debitis iste. Ea impedit ullam reiciendis ut, suscipit beatae nostrum fugiat sapiente quaerat quam adipisci assumenda, earum nisi ab, illum doloremque unde molestias velit similique est. Architecto iste ut numquam voluptatibus quod et excepturi, possimus quisquam ullam deserunt totam illo optio sunt! Omnis nam quis quo beatae esse. Perspiciatis enim magni, amet facilis beatae unde earum natus numquam, dolorum eum blanditiis corporis quisquam incidunt voluptates similique. Minus quia architecto tempora dignissimos dolores sit eligendi temporibus provident qui omnis, eius quas, repudiandae, doloribus hic.

And Some More

Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque, dicta natus? Mollitia laborum voluptatem consequatur nihil ipsam debitis iste. Ea impedit ullam reiciendis ut, suscipit beatae nostrum fugiat sapiente quaerat quam adipisci assumenda, earum nisi ab, illum doloremque unde molestias velit similique est. Architecto iste ut numquam voluptatibus quod et excepturi, possimus quisquam ullam deserunt totam illo optio sunt! Omnis nam quis quo beatae esse. Perspiciatis enim magni, amet facilis beatae unde earum natus numquam, dolorum eum blanditiis corporis quisquam incidunt voluptates similique. Minus quia architecto tempora dignissimos dolores sit eligendi temporibus provident qui omnis, eius quas, repudiandae, doloribus hic.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque, dicta natus? Mollitia laborum voluptatem consequatur nihil ipsam debitis iste. Ea impedit ullam reiciendis ut, suscipit beatae nostrum fugiat sapiente quaerat quam adipisci assumenda, earum nisi ab, illum doloremque unde molestias velit similique est. Architecto iste ut numquam voluptatibus quod et excepturi, possimus quisquam ullam deserunt totam illo optio sunt! Omnis nam quis quo beatae esse. Perspiciatis enim magni, amet facilis beatae unde earum natus numquam, dolorum eum blanditiis corporis quisquam incidunt voluptates similique. Minus quia architecto tempora dignissimos dolores sit eligendi temporibus provident qui omnis, eius quas, repudiandae, doloribus hic.

© Copyright 2019- QA ltd.



Opening Hours

	Opens	Closes
Monday	11:00	00:00
Friday	11:00	2:00
Sunday	11:44	23:00
Tuesday	14:30	18:00

What's On

Click on a Film to see more

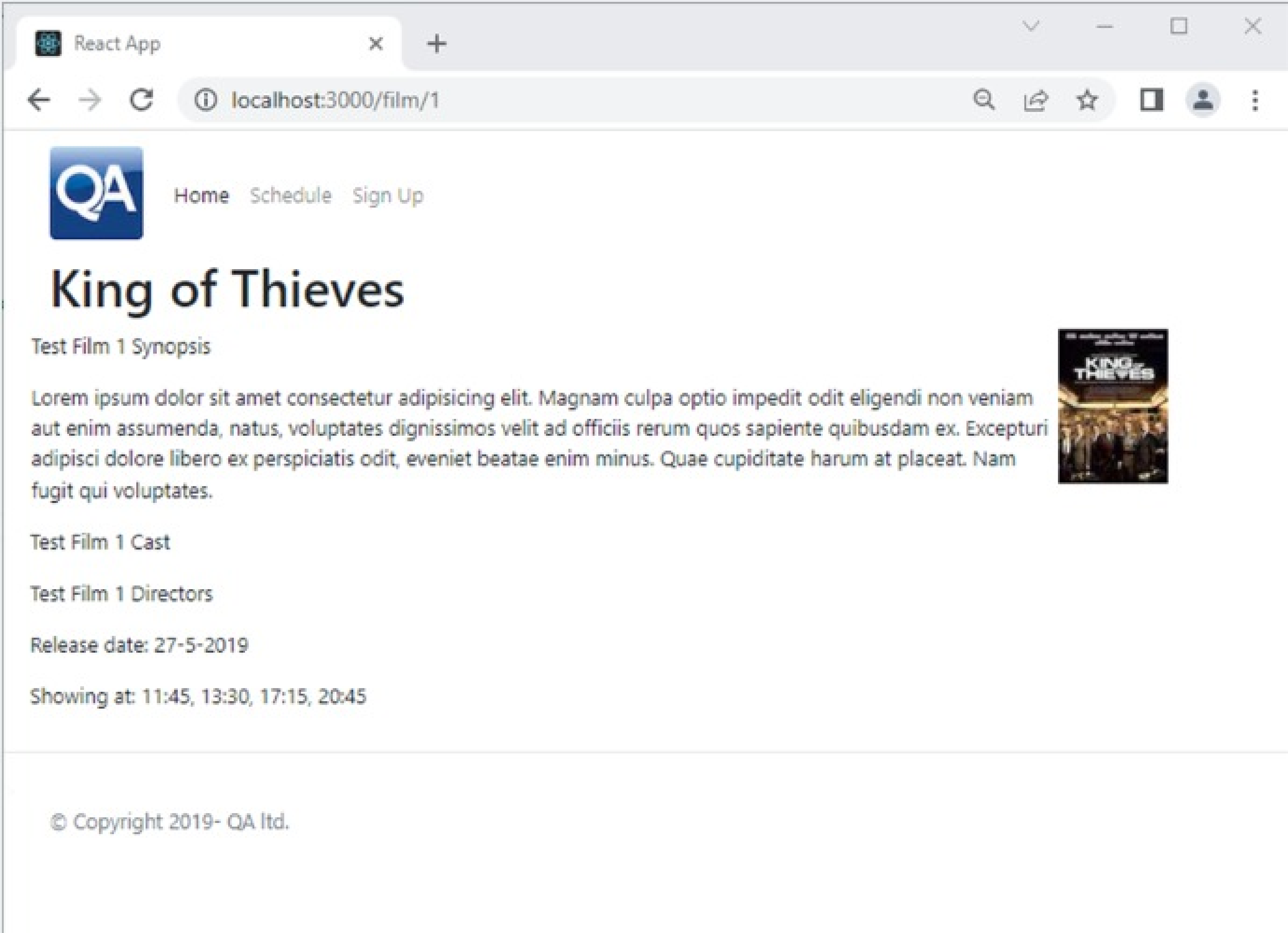
Release Date	Show Times	Title	Image
27-5-2019	11:45, 13:30, 17:15, 20:45	King of Thieves	
27-10-2019	11:45, 13:30, 17:15, 20:45	The Predator	
3-5-2022	9:25, 12:30	The Banana Men	

Coming Soon

Click on a Film to see more

Release Date	Title	Image
27-5-2019	The House with a Clock in its Walls	

© Copyright 2019- QA ltd.



King of Thieves

Test Film 1 Synopsis

Lorem ipsum dolor sit amet consectetur adipisicing elit. Magnam culpa optio impedit odit eligendi non veniam aut enim assumenda, natus, voluptates dignissimos velit ad officiis rerum quos sapiente quibusdam ex. Excepturi adipisci dolore libero ex perspiciatis odit, eveniet beatae enim minus. Quae cupiditate harum at placeat. Nam fugit qui voluptates.

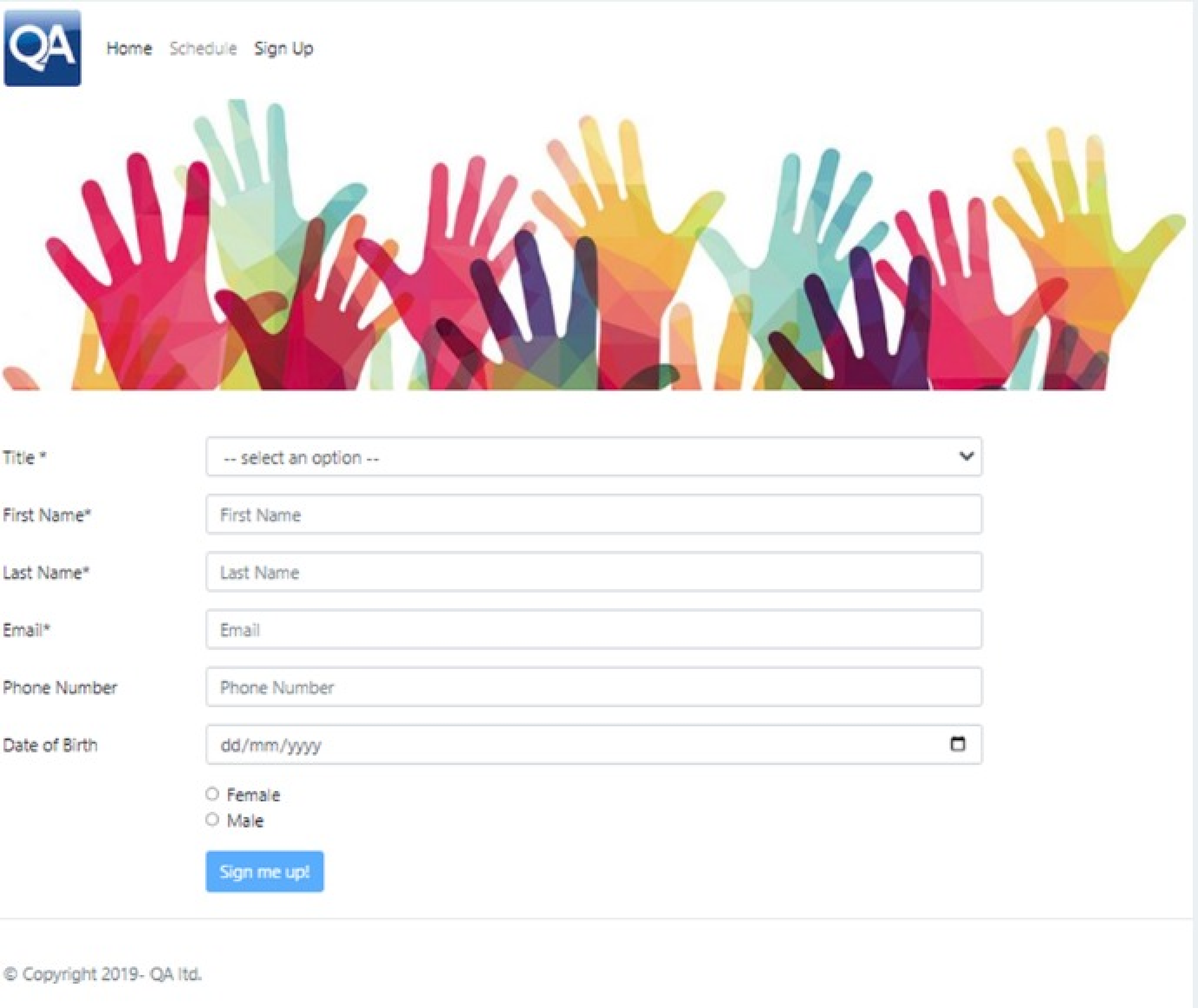
Test Film 1 Cast

Test Film 1 Directors

Release date: 27-5-2019

Showing at: 11:45, 13:30, 17:15, 20:45

© Copyright 2019- QA ltd.



Title *	-- select an option --
First Name*	First Name
Last Name*	Last Name
Email*	Email
Phone Number	Phone Number
Date of Birth	dd/mm/yyyy

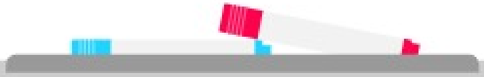
○ Female
○ Male

THE C# PROGRAMMING LANGUAGE

Apply stage:

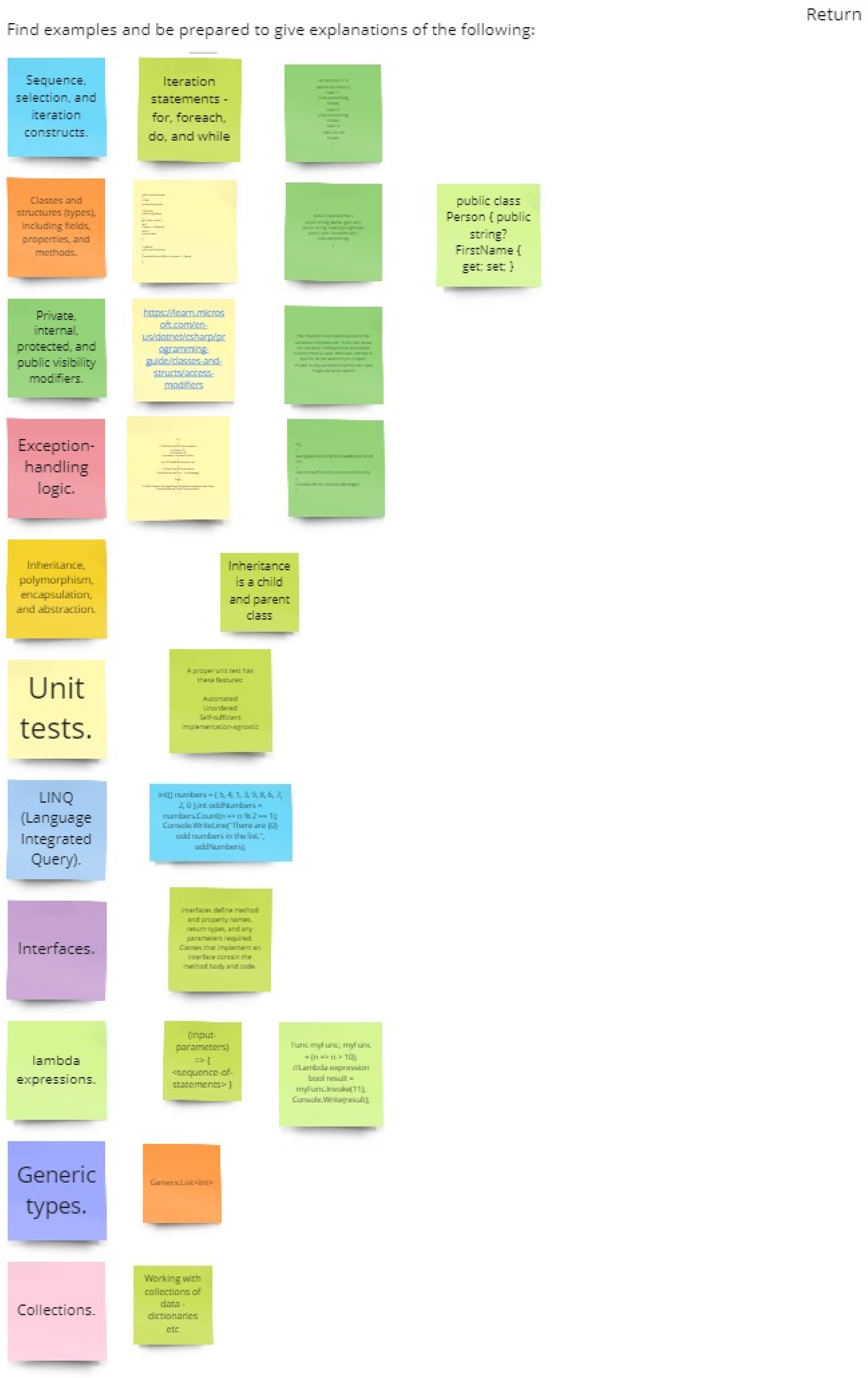
Workplace application plan

Any questions?



- Review all questions in the Apply Stage documentation.
- Ensure you have completed all of the relevant documents.
- If you haven't done so already, discuss your plan with your line manager.
- Complete the Apply work.
- Reflect on your experience.
- Once you have completed all the required steps confirm this has been done and have this validated by your line manager.
- Finally, submit your work.
- We will then check your submission, mark your activity as complete and issue you with your **C# Impact Badge**.

Good luck!



Return

Find examples and be prepared to give explanations of the following:

Sequence, selection, and iteration constructs.

Iteration statements - for, foreach, do, and while

[illegible]

Classes and structures (types), including fields, properties, and methods.

[illegible]

[illegible]

public class Person { public string? FirstName { get; set; }

Private, internal, protected, and public visibility modifiers.

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers

[illegible]

Exception-handling logic.

[illegible]

[illegible]

Inheritance, polymorphism, encapsulation, and abstraction.

Inheritance is a child and parent class

Unit tests.

A proper unit test has these features:

Automated
Unordered
Self-sufficient
Implementation-agnostic

LINQ (Language Integrated Query).

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };int oddNumbers = numbers.Count(n => n % 2 == 1); Console.WriteLine("There are {0} odd numbers in the list.", oddNumbers);
```

Interfaces.

Interfaces define method and property names, return types, and any parameters required. Classes that implement an interface contain the method body and code.

lambda expressions.

(input-parameters) => { <sequence-of-statements> }

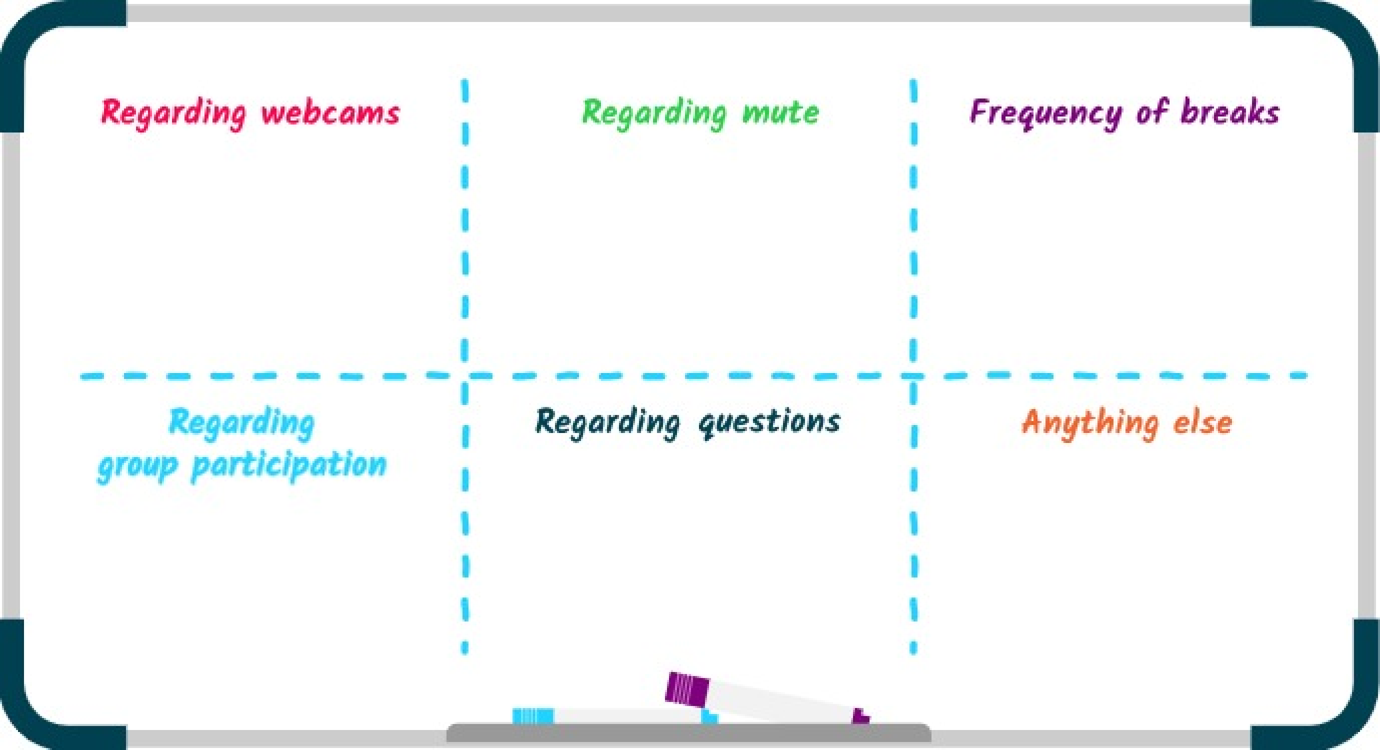
```
Func myFunc; myFunc = (n => n > 10); //Lambda expression bool result = myFunc.Invoke(11); Console.Write(result);
```

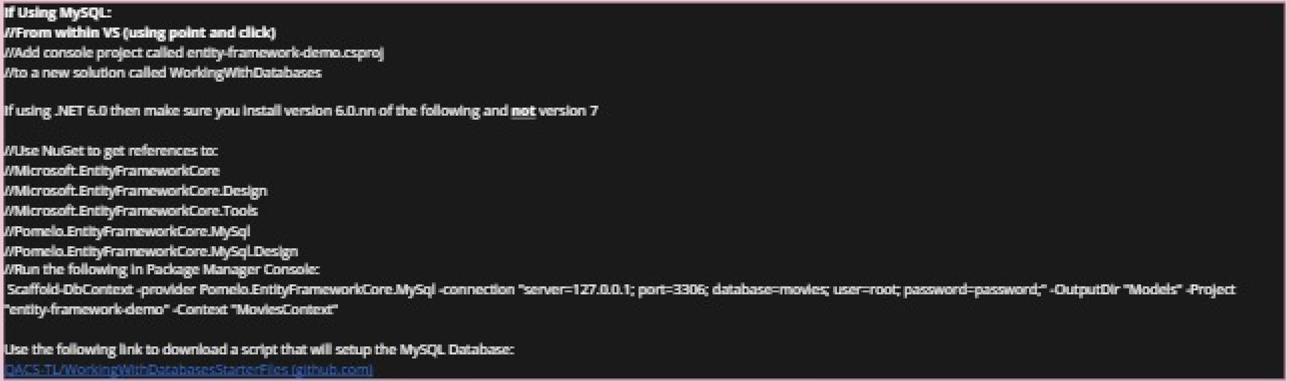
Generic types.

GenericList<int>

Collections.

Working with collections of data - dictionaries etc


Regarding webcams
Regarding mute
Frequency of breaks
Regarding group participation
Regarding questions
Anything else



If Using MySQL:
//From within VS (using point and click)
//Add console project called entity-framework-demo.csproj
//to a new solution called WorkingWithDatabases

If using .NET 6.0 then make sure you install version 6.0.nn of the following and not version 7

```
//Use NuGet to get references to:
//Microsoft.EntityFrameworkCore
//Microsoft.EntityFrameworkCore.Design
//Microsoft.EntityFrameworkCore.Tools
//Pomelo.EntityFrameworkCore.MySql
//Pomelo.EntityFrameworkCore.MySql.Design
//Run the following in Package Manager Console:
 Scaffold-DbContext -provider Pomelo.EntityFrameworkCore.MySql -connection "server=127.0.0.1; port=3306; database=movies; user=root; password=password;" -OutputDir "Models" -Project "entity-framework-demo" -Context "MoviesContext"
```

Use the following link to download a script that will setup the MySQL Database:
QACS-TL/WorkingWithDatabasesStarterFiles (github.com)

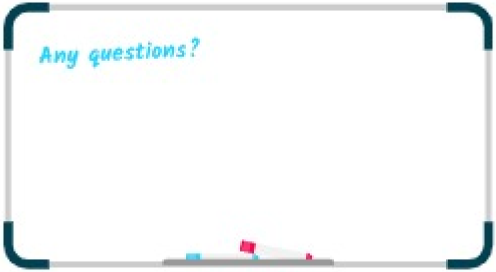
Click here for Solutions

Apply Workshop

THE C#
PROGRAMMING
LANGUAGE

Apply stage:

Workplace application plan



- Review all questions in the Apply Stage documentation.
- Ensure you have completed all of the relevant documents.
- If you haven't done so already, discuss your plan with your line manager.
- Complete the Apply work.
- Reflect on your experience.
- Once you have completed all the required steps confirm this has been done and have this validated by your line manager.
- Finally, submit your work.
- We will then check your submission, mark your activity as complete and issue you with your **C# Impact Badge**.

Good luck!