Web API assignment

Requirement

- Implement a web API that converts the long URL to the short URL. For example, the original URL is https://www.finning.com/welcome/canada/monitor/validate. The output URL will be https://example.co/ysrAXm.
- The based URL should be configurable. The default value is https://example.co. Please see the example below.
- The maximum length of the random string should be configurable. The default value is 6. For example, the arbitrary string in the output URL is `ysrAXm`. The string should be generated randomly and encrypted.
- Throw the argument exception if the original URL string is empty or null.
- Throw the incorrect format exception if the original URL format is invalid.
- Implement the unit tests.
- Write a brief document of the CI/CD pipeline configuration and deployment plans.
- In the document, share some thoughts on what gaps you see in the requirements or your current solutions and how you would address those if this were a real production service
- No UI needs to be implemented.
- No authentication needs to be implemented.
- Use the provided cryptography class to generate the short URL.
- Keep code as simple as possible.

Sample code

App Settings

```
{
  "ShrinkUrlSettings": {
    "BaseUrl": "https://example.co/",
    "MaxLength": 6
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Cryptography class

```
public static class Cryptography
    {
        private const string ENCRYPTION_KEY = "A60934D8C1A2AC3A69642A3902198";
        private readonly static byte[] SALT = new byte[] { 99, 52, 2, 24, 51, 67,
22, 88 };

        public static string EncryptUrl(string originalUrl, int maxLength)
        {
            byte[] plainText = Encoding.Unicode.GetBytes(originalUrl);
            var secretKey = new Rfc2898DeriveBytes(ENCRYPTION_KEY, SALT);

            using (RijndaelManaged rijndaelCipher = new RijndaelManaged())
            {
                using (ICryptoTransform encryptor =
rijndaelCipher.CreateEncryptor(secretKey.GetBytes(32), secretKey.GetBytes(16)))
                using (var memoryStream = new MemoryStream())
                using (var cryptoStream = new CryptoStream(memoryStream,
encryptor, CryptoStreamMode.Write))
                {
                    cryptoStream.Write(plainText, 0, plainText.Length);
                    cryptoStream.FlushFinalBlock();
                    string encryptedUrl =
Convert.ToBase64String(memoryStream.ToArray());
                    encryptedUrl = HttpUtility.UrlEncode(encryptedUrl);

                    if (encryptedUrl.Length > maxLength)
                    {
                        encryptedUrl = encryptedUrl.Substring(0, maxLength);
                    }
                    return encryptedUrl;
                }
            }
        }
    }
```