Lab Manual for Cloud Computing

Lab No. 13

Two-Factor Authentication using Google Authenticator in Asp.Net MVC

LAB 13: TWO-FACTOR AUTHENTICATION USING GOOGLE AUTHENTICATOR IN ASP.NET MVC

1. INTRODUCTION:

Introduction to Two-Factor Authentication (2FA) using Google Authenticator in ASP.NET MVC:

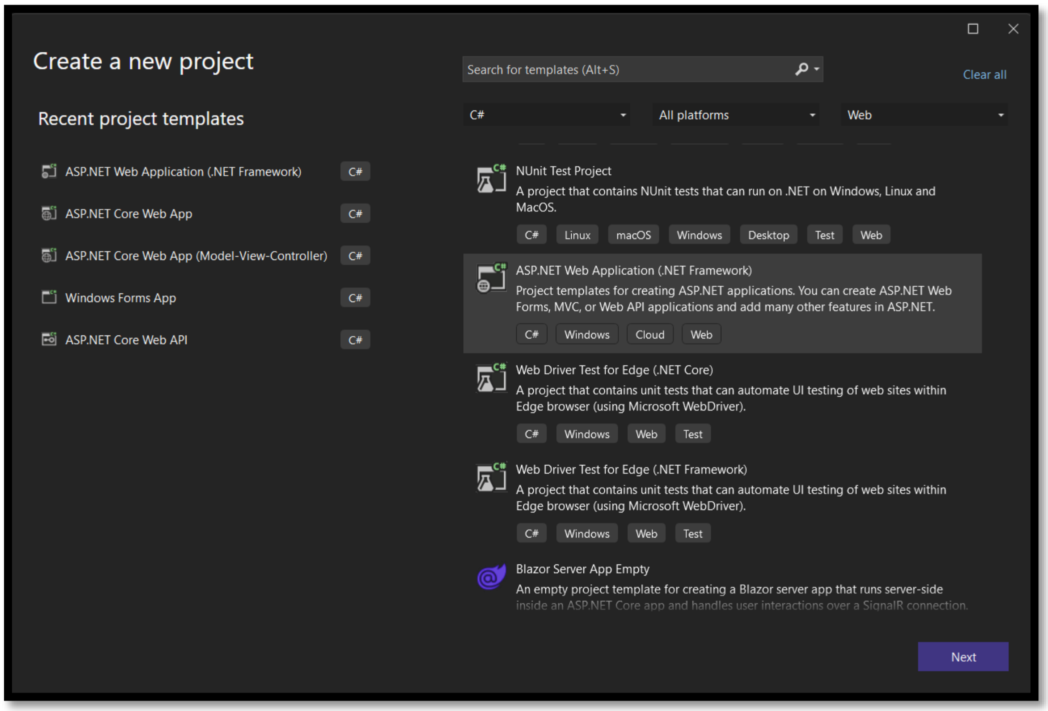
In the realm of cybersecurity, Two-Factor Authentication (2FA) stands as a critical defense mechanism against unauthorized access. By requiring users to provide two forms of verification, typically a password and a temporary code, 2FA adds an extra layer of security to account authentication. In ASP.NET MVC applications, integrating 2FA with Google Authenticator emerges as a robust solution, leveraging time-based one-time passwords (TOTP) generated by users' mobile devices. This method enhances security by necessitating not only something users know (their password) but also something they possess (their mobile device), effectively thwarting various threats such as phishing attacks and password breaches.

Usage and Need for Two-Factor Authentication with Google Authenticator in ASP.NET MVC:

Implementing 2FA with Google Authenticator in ASP.NET MVC bolsters security measures while aligning with industry standards and compliance requirements. It mitigates risks associated with password-based authentication by providing an additional verification step, thereby safeguarding sensitive user data and bolstering user trust. Furthermore, the flexible implementation and user-friendly nature of Google Authenticator make it an ideal choice for enhancing security without introducing significant complexities for users, thus ensuring a seamless and secure authentication experience within ASP.NET MVC applications.

Example:

Step 1: Create a new project in Visual Studio, select an **ASP.NET Web Application (.Net Framework),** and press the Next button.


Create a new project

Recent project templates
- ASP.NET Web Application (.NET Framework) C#
- ASP.NET Core Web App C#
- ASP.NET Core Web App (Model-View-Controller) C#
- Windows Forms App C#
- ASP.NET Core Web API C#

Search for templates (Alt+S) | Clear all

C# | All platforms | Web

NUnit Test Project
A project that contains NUnit tests that can run on .NET on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

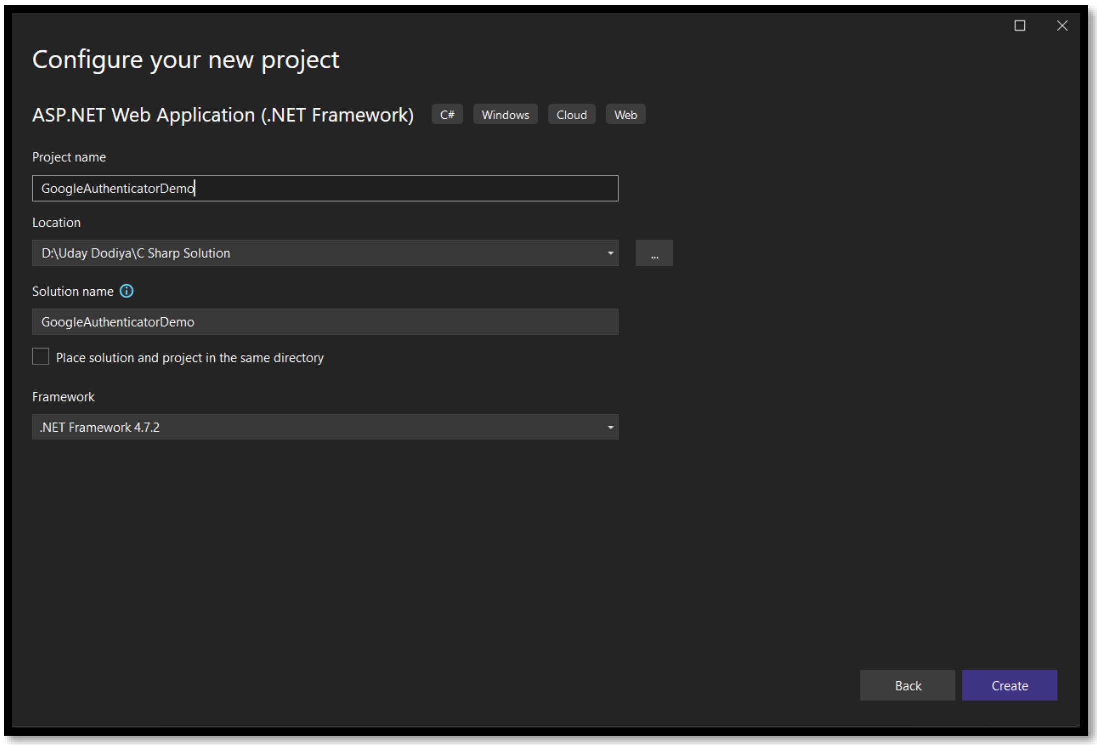
Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Web Driver Test for Edge (.NET Framework)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Blazor Server App Empty
An empty project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection.

Next

Step 2: Give the project a name, select the save location folder, and click on the Create button.



Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name: GoogleAuthenticatorDemo

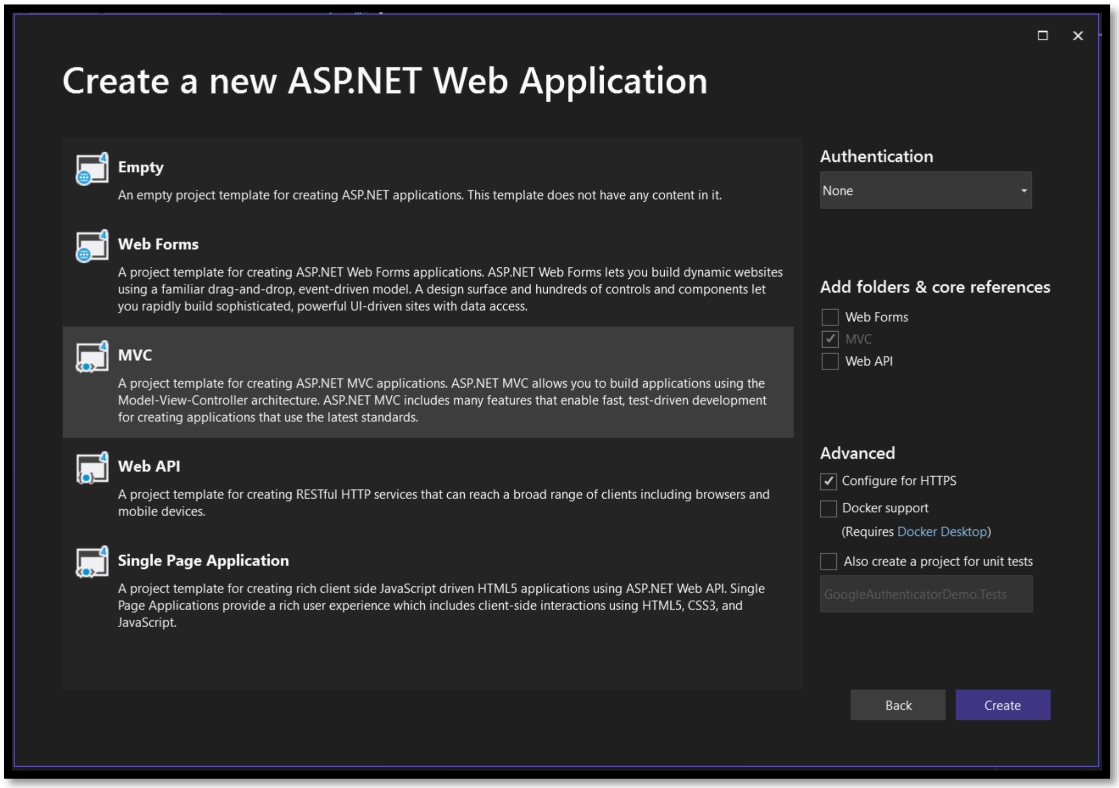
Location: D:\Uday Dodiya\C Sharp Solution

Solution name: GoogleAuthenticatorDemo

☐ Place solution and project in the same directory

Framework: .NET Framework 4.7.2

Step 3: Select **MVC** Option for project type and click on the Create button. Now your project is created.



Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

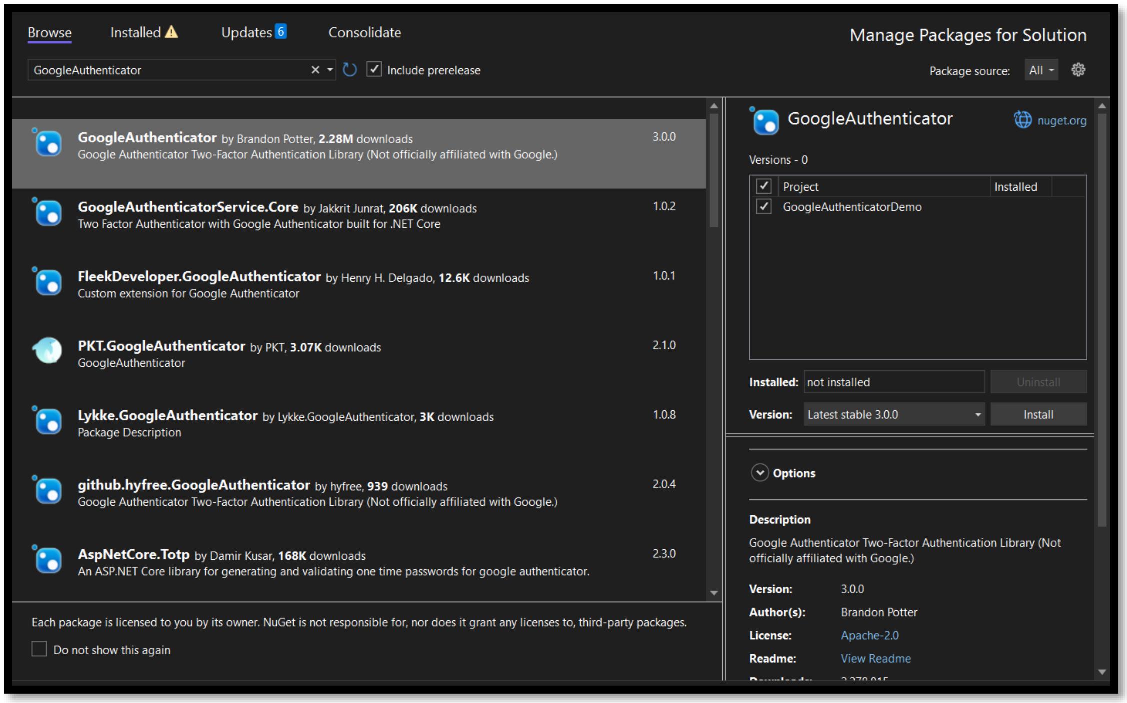
Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication: None

Add folders & core references
☐ Web Forms
☑ MVC
☐ Web API

Advanced
☑ Configure for HTTPS
☐ Docker support
(Requires Docker Desktop)
☐ Also create a project for unit tests
GoogleAuthenticatorDemo.Tests

Step 4: Add the Google.Authenticator package from NuGet. Go to Tools Menu - **NuGet Package Manager** - Manage NuGetPackages For Solution - Search for **Google.Authenticator** > Install > Close.



Browse | Installed | Updates 6 | Consolidate

Manage Packages for Solution

GoogleAuthenticator ☑ Include prerelease

Package source: All

Package	Version
GoogleAuthenticator by Brandon Potter, **2.28M** downloads Google Authenticator Two-Factor Authentication Library (Not officially affiliated with Google.)	3.0.0
GoogleAuthenticatorService.Core by Jakkrit Junrat, **206K** downloads Two Factor Authenticator with Google Authenticator built for .NET Core	1.0.2
FleekDeveloper.GoogleAuthenticator by Henry H. Delgado, **12.6K** downloads Custom extension for Google Authenticator	1.0.1
PKT.GoogleAuthenticator by PKT, **3.07K** downloads GoogleAuthenticator	2.1.0
Lykke.GoogleAuthenticator by Lykke.GoogleAuthenticator, **3K** downloads Package Description	1.0.8
github.hyfree.GoogleAuthenticator by hyfree, **939** downloads Google Authenticator Two-Factor Authentication Library (Not officially affiliated with Google.)	2.0.4
AspNetCore.Totp by Damir Kusar, **168K** downloads An ASP.NET Core library for generating and validating one time passwords for google authenticator.	2.3.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

GoogleAuthenticator nuget.org

Versions - 0

☑ Project	Installed
☑ GoogleAuthenticatorDemo	

Installed: not installed Uninstall

Version: Latest stable 3.0.0 Install

Options

Description

Google Authenticator Two-Factor Authentication Library (Not officially affiliated with Google.)

Version:	3.0.0
Author(s):	Brandon Potter
License:	Apache-2.0
Readme:	View Readme

Step 5: Add the Model class **LoginModel.cs** under **Model Folder** for login details that stores the username and password.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace GoogleAuthenticatorDemo.Models
{
    public class LoginModel
    {
        public string UserName { get; set; }

        public string Password { get; set; }
    }
}
```

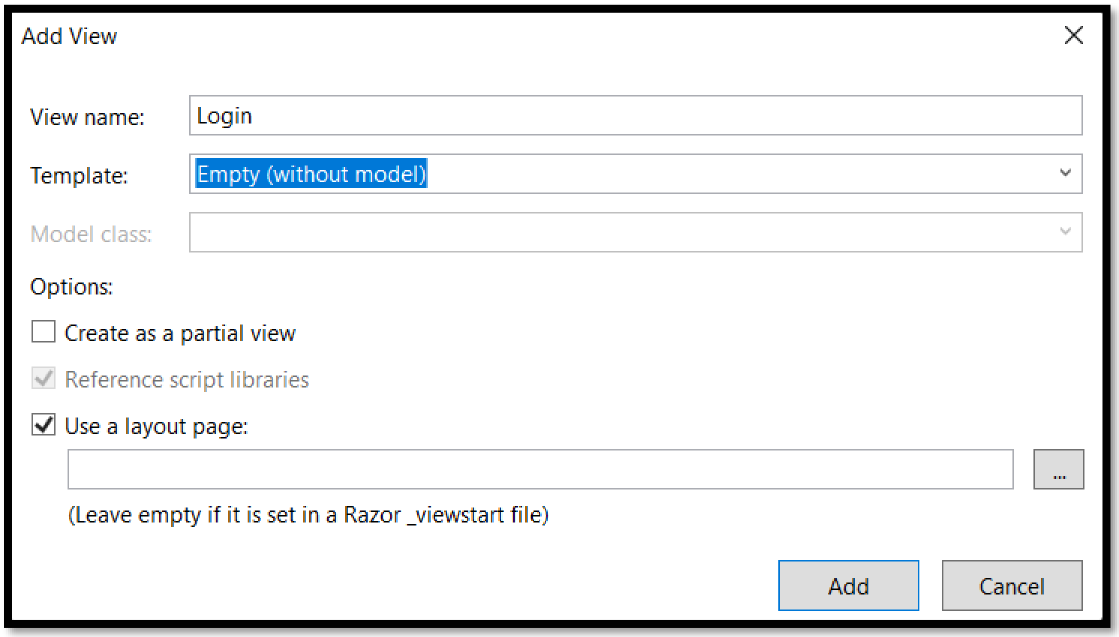
Step 6: Add the Google Authentication Private Key In **Web.Config** File under <appSettings>.This key is not fixed; you can create it yourself with any character combination.Any character combination use as private key in google authenticator.

```
<appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="GoogleAuthKey" value="Noo78fdj" />
</appSettings>
```

Step 7: Create a **Login action** method in your controller; in my case, it's HomeController.

```
public ActionResult Login()
{
    Session["UserName"] = null;
    Session["IsValidTwoFactorAuthentication"] = null;
    return View();
}
```

Step 8: Right click on the **Login action** method and select **Add View Option**, and then select the following: Then click the Add Button button to add a view for this action method.



Add View

View name: Login

Template: Empty (without model)

Model class:

Options:

☐ Create as a partial view

✓ Reference script libraries

✓ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Now a new file is created under View > Home > Login.cshtml.

Step 9: Add the following code into **Login.cshtml:**

```
@model GoogleAuthenticatorDemo.Models.LoginModel
@{
    ViewBag.Title = "Login";
}

<center>
    <h2>Login Page</h2>
    @if (ViewBag.Status == null || !ViewBag.Status)
    {
        <div>@ViewBag.Message</div>
        <div>
            @using (Html.BeginForm())
            {
                <div class="form-group">
                    <label for="UserName">UserName : </label>
                    @Html.TextBoxFor(a => a.UserName, new { @class = "form-control" })
                </div>
                <div class="form-group">
                    <label for="Password">Password : </label>
                    @Html.TextBoxFor(a => a.Password, new { @class = "form-control",
                                                   type = "password" })
                </div>
                <input type="submit" value="Login" class="btn btn-default" />
            }
        </div>
    }
    else
    {

        <div>@ViewBag.Message</div>
        <div>
            <img src="@ViewBag.BarcodeImageUrl" width="300" height="300" />

        </div>
        <div>
            Manual Setup Code : @ViewBag.SetupCode
        </div>
        <div>
            @using (Html.BeginForm("TwoFactorAuthenticate", "Home", FormMethod.Post))
            {
                <input type="text" name="CodeDigit" />
                <input type="submit" class="btn btn-success" />
            }
        </div>
    }
</center>
```

Step 10: Create a **Post** type **Login Action method** in your controller and add the following code:

```
[HttpPost]
public ActionResult Login(LoginModel login)
{
  bool status = false;

  if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] == null ||
!(bool) Session["IsValidTwoFactorAuthentication"])
  {
      string googleAuthKey = WebConfigurationManager.AppSettings["GoogleAuthKey"];
      string UserUniqueKey = (login.UserName + googleAuthKey);

      //Take UserName And Password As Static - Admin As User And 12345 As Password
      if (login.UserName == "Admin" && login.Password == "12345")
      {
         Session["UserName"] = login.UserName;

         //Two Factor Authentication Setup
         TwoFactorAuthenticator TwoFacAuth = new TwoFactorAuthenticator();
         var setupInfo = TwoFacAuth.GenerateSetupCode("UdayDodiyaAuthDemo.com",
         login.UserName, ConvertSecretToBytes(UserUniqueKey, false), 300);
         Session["UserUniqueKey"] = UserUniqueKey;
         ViewBag.BarcodeImageUrl = setupInfo.QrCodeSetupImageUrl;
         ViewBag.SetupCode = setupInfo.ManualEntryKey;
         status = true;
      }
  }
  else
  {
    return RedirectToAction("Index");
  }
  ViewBag.Status = status;
  return View();
}
```

Step 11: Create the **ConvertSecretToBytes** function in the controller.

```
private static byte[] ConvertSecretToBytes(string secret, bool secretIsBase32) =>
           secretIsBase32 ? Base32Encoding.ToBytes(secret) : Encoding.UTF8.GetBytes(secret);
```

Step 12: Create a **TwoFactorAuthenticate** Action action method for your controller.

```
public ActionResult TwoFactorAuthenticate()
{
     var token = Request["CodeDigit"];
     TwoFactorAuthenticator TwoFacAuth = new TwoFactorAuthenticator();
     string UserUniqueKey = Session["UserUniqueKey"].ToString();
     bool isValid = TwoFacAuth.ValidateTwoFactorPIN(UserUniqueKey, token, false);
     if (isValid)
     {
         HttpCookie TwoFCookie = new HttpCookie("TwoFCookie");
         string UserCode
=Convert.ToBase64String(MachineKey.Protect(Encoding.UTF8.GetBytes(UserUniqueKey)));

         Session["IsValidTwoFactorAuthentication"] = true;
         return RedirectToAction("Index");
      }

      ViewBag.Message = "Google Two Factor PIN is expired or wrong";
      return RedirectToAction("Login");
}
```

Step 13: Create a **Logoff** Action Method in your controller to handle the logout action.

```
public ActionResult Logoff()
{
     Session["UserName"] = null;
     Session["IsValidTwoFactorAuthentication"] = null;
     return RedirectToAction("Login");
}
```

Step 14: To handle an **unauthorised login**, add the following condition to all of your action methods, which in my case are three method.

```
public ActionResult Index()
{
    if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] == null ||
!(bool)Session["IsValidTwoFactorAuthentication"])
    {
        return RedirectToAction("Login");
    }
    return View();
}

public ActionResult About()
{
     if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] == null ||
!(bool)Session["IsValidTwoFactorAuthentication"])
     {
         return RedirectToAction("Login");
     }

     ViewBag.Message = "Your application description page.";
     return View();
}

public ActionResult Contact()
{
     if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] == null ||
!(bool)Session["IsValidTwoFactorAuthentication"])
     {
         return RedirectToAction("Login");
     }

     ViewBag.Message = "Your contact page.";
     return View();
}
```

Final HomeController.cs File

```
using Google.Authenticator;
using GoogleAuthenticatorDemo.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Configuration;
using System.Web.Mvc;
using System.Web.Security;

namespace GoogleAuthenticatorDemo.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] ==
null || !(bool)Session["IsValidTwoFactorAuthentication"])
            {
                return RedirectToAction("Login");
            }
            return View();
        }

        public ActionResult About()
        {
            if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] ==
null || !(bool)Session["IsValidTwoFactorAuthentication"])
            {
                return RedirectToAction("Login");
            }

            ViewBag.Message = "Your application description page.";
            return View();
        }

        public ActionResult Contact()
        {
            if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] ==
null || !(bool)Session["IsValidTwoFactorAuthentication"])
            {
                return RedirectToAction("Login");
            }

            ViewBag.Message = "Your contact page.";
            return View();
        }

        public ActionResult Login()
        {
            Session["UserName"] = null;
            Session["IsValidTwoFactorAuthentication"] = null;
            return View();
        }

        [HttpPost]
        public ActionResult Login(LoginModel login)
        {
            bool status = false;

            if (Session["Username"] == null || Session["IsValidTwoFactorAuthentication"] ==
null || !(bool)Session["IsValidTwoFactorAuthentication"])
            {
                string googleAuthKey = WebConfigurationManager.AppSettings["GoogleAuthKey"];
                string UserUniqueKey = (login.UserName + googleAuthKey);


                //Take UserName And Password As Static - Admin As User And 12345 As Password
                if (login.UserName == "Admin" && login.Password == "12345")
                {
                    Session["UserName"] = login.UserName;

                    //Two Factor Authentication Setup
                    TwoFactorAuthenticator TwoFacAuth = new TwoFactorAuthenticator();
                    var setupInfo = TwoFacAuth.GenerateSetupCode("Googleauthenticator.com",
login.UserName, ConvertSecretToBytes(UserUniqueKey, false), 300);
                    Session["UserUniqueKey"] = UserUniqueKey;
                    ViewBag.BarcodeImageUrl = setupInfo.QrCodeSetupImageUrl;
                    ViewBag.SetupCode = setupInfo.ManualEntryKey;
                    status = true;
                }
            }
            else
            {
                return RedirectToAction("Index");
            }
            ViewBag.Status = status;
            return View();
        }


        private static byte[] ConvertSecretToBytes(string secret, bool secretIsBase32) =>
           secretIsBase32 ? Base32Encoding.ToBytes(secret) : Encoding.UTF8.GetBytes(secret);


        public ActionResult TwoFactorAuthenticate()
        {
            var token = Request["CodeDigit"];
            TwoFactorAuthenticator TwoFacAuth = new TwoFactorAuthenticator();
            string UserUniqueKey = Session["UserUniqueKey"].ToString();
            bool isValid = TwoFacAuth.ValidateTwoFactorPIN(UserUniqueKey, token, false);
            if (isValid)
            {
                HttpCookie TwoFCookie = new HttpCookie("TwoFCookie");
                string UserCode =
Convert.ToBase64String(MachineKey.Protect(Encoding.UTF8.GetBytes(UserUniqueKey)));

                Session["IsValidTwoFactorAuthentication"] = true;
                return RedirectToAction("Index");
            }
            ViewBag.Message = "Google Two Factor PIN is expired or wrong";
            return RedirectToAction("Login");
        }

        public ActionResult Logoff()
        {
            Session["UserName"] = null;
            Session["IsValidTwoFactorAuthentication"] = null;
            return RedirectToAction("Login");
        }
    }
}
```

Step 15: All pages display the user's name and the option to log out. So add that code to the **_Layout.cshtml** page, and your _Layout.cshtml should now look like this:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse" title="more options">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new {
@class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>

                @if (@HttpContext.Current.Session["UserName"] != null &&
@HttpContext.Current.Session["IsValidTwoFactorAuthentication"] != null)
                {

if(bool.Parse(@HttpContext.Current.Session["IsValidTwoFactorAuthentication"].ToString()) ==
true)
                   {
                        <ul class="nav navbar-nav navbar-right">
                            <li style="color: White; font-size: 15px; margin-top: 10px; margin-
right: 100px;">
                                USER: @HttpContext.Current.Session["UserName"].ToString()
                            </li>
                            <li>
                                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                @Html.ActionLink("Log Out", "Logoff", "Home", new { style =
"color:White; font-size: 15px" })
                            </li>
                        </ul>
                   }
                }
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

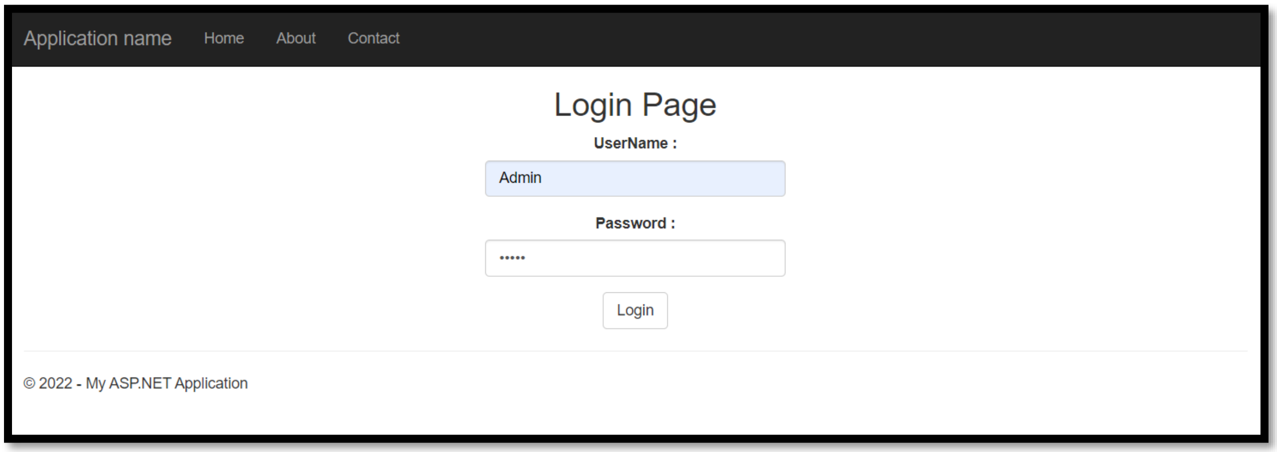
Step 16: Display the username on the index page. So, add the following code to your **index.cshtml** page, and your index.cshtml should now look like this:

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>ASP.NET</h1>
    <p class="lead">ASP.NET is a free web framework for building great Web sites and Web
applications using HTML, CSS and JavaScript.</p>
    <p><a href="https://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
```

```
</div>
<center><h2 style="color: red;">Welcome Mr.
@HttpContext.Current.Session["UserName"].ToString()</h2></center>
<div class="row">
    <div class="col-md-4">
        <h2>Getting started</h2>

        <p>
            ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that
            enables a clean separation of concerns and gives you full control over markup
            for enjoyable, agile development.
        </p>
        <p><a class="btn btn-default"
href="https://go.microsoft.com/fwlink/?LinkId=301865">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Get more libraries</h2>
        <p>NuGet is a free Visual Studio extension that makes it easy to add, remove, and
update libraries and tools in Visual Studio projects.</p>
        <p><a class="btn btn-default"
href="https://go.microsoft.com/fwlink/?LinkId=301866">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Web Hosting</h2>
        <p>You can easily find a web hosting company that offers the right mix of features and
price for your applications.</p>
        <p><a class="btn btn-default"
href="https://go.microsoft.com/fwlink/?LinkId=301867">Learn more &raquo;</a></p>
    </div>
</div>
```

You can design your page yourself and according to your needs, but I've used default content, only adding a username to understand the concept.

Step 17: Now run your project. When you run the project, by default, the index method will invoke but not authorize, so it will redirect to the login page. Enter "**Admin**" as the user name and "**12345**" as the password, then click the login button.



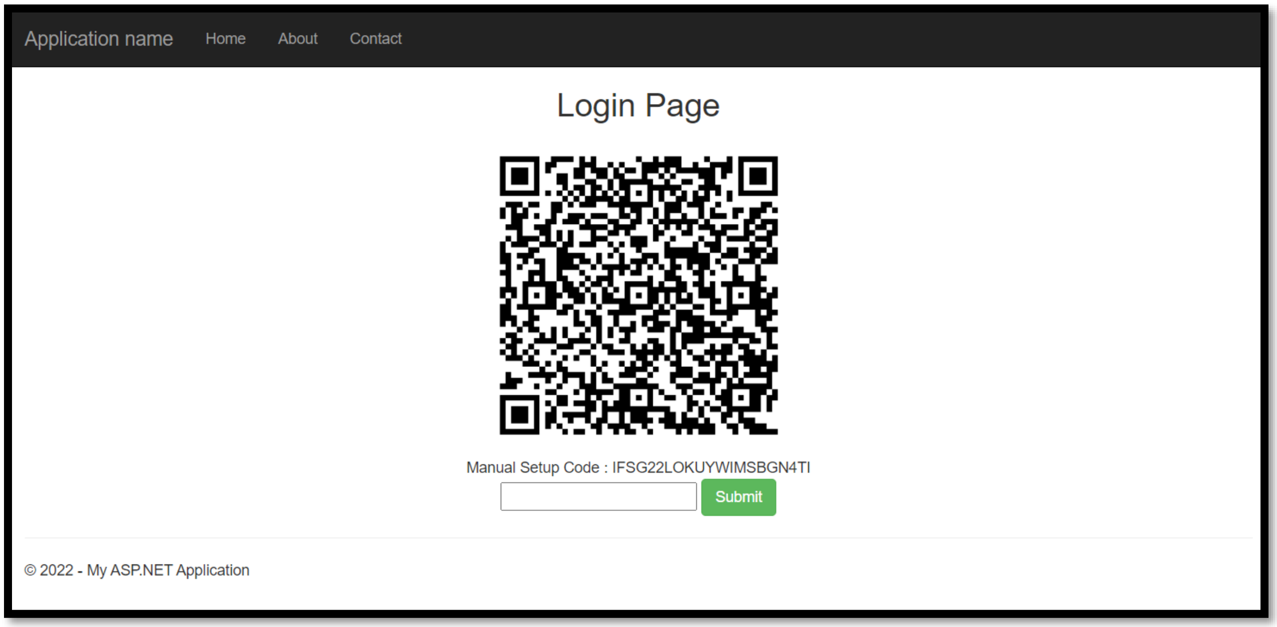
Login Page

UserName : Admin

Password : •••••

© 2022 - My ASP.NET Application

Step 18: After clicking the login button, a **QR code** and the **setup key** for two-factor authentication were generated, which will be used later on the mobile app for code generation.

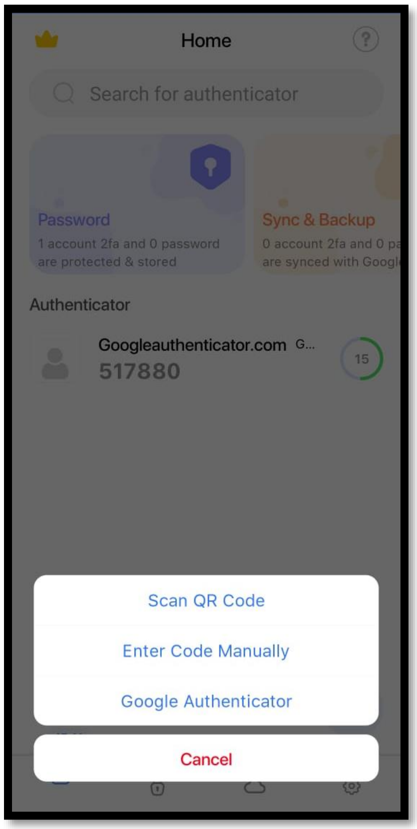


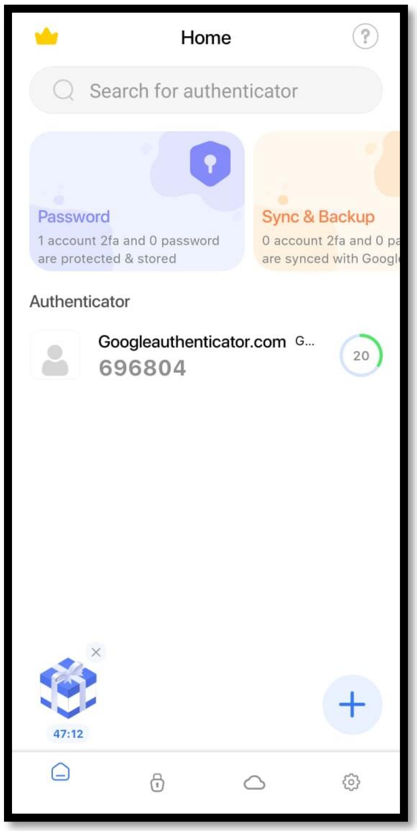
Login Page

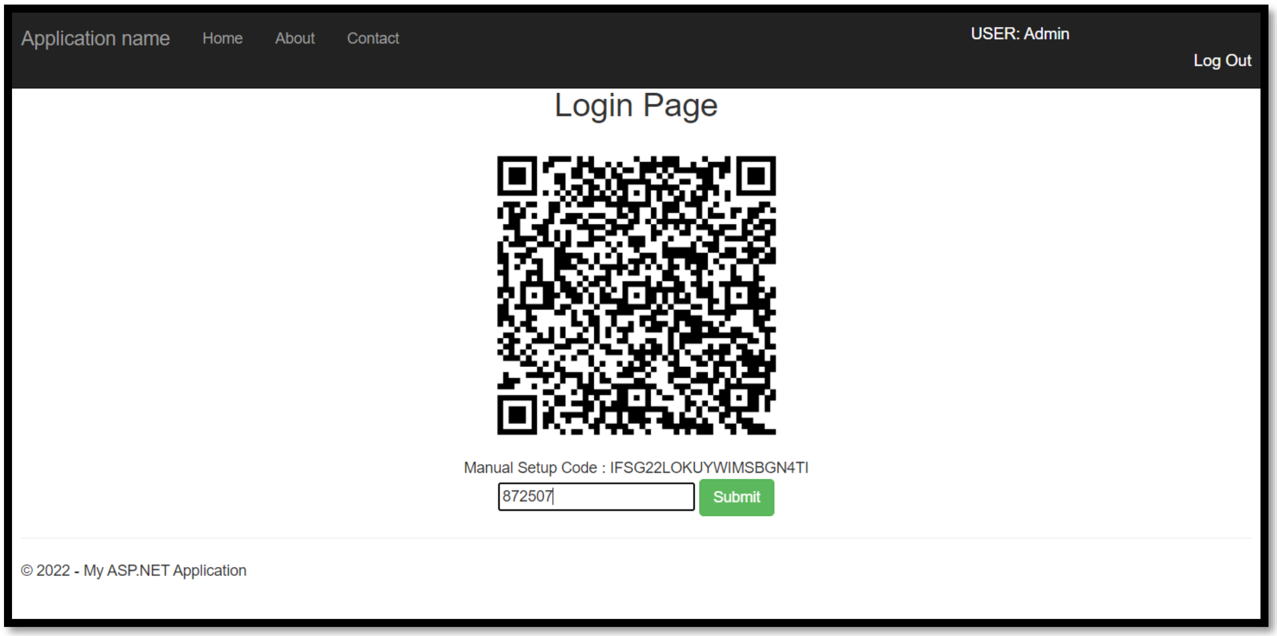
Manual Setup Code : IFSG22LOKUYWIMSBGN4TI

© 2022 - My ASP.NET Application

Step 19: Install the **Google Authenticator** app from the Play Store or here on your mobile device, and open it; you have two options: scan a QR code or enter a setup key. We scan the QR code here, so you only have one Authenticator Code, which will change after a while. Enter this code on your website.




Home
Search for authenticator
Password
1 account 2fa and 0 password are protected & stored
Sync & Backup
0 account 2fa and 0 pa are synced with Googl
Authenticator
Googleauthenticator.com G...
696804
20
47:12



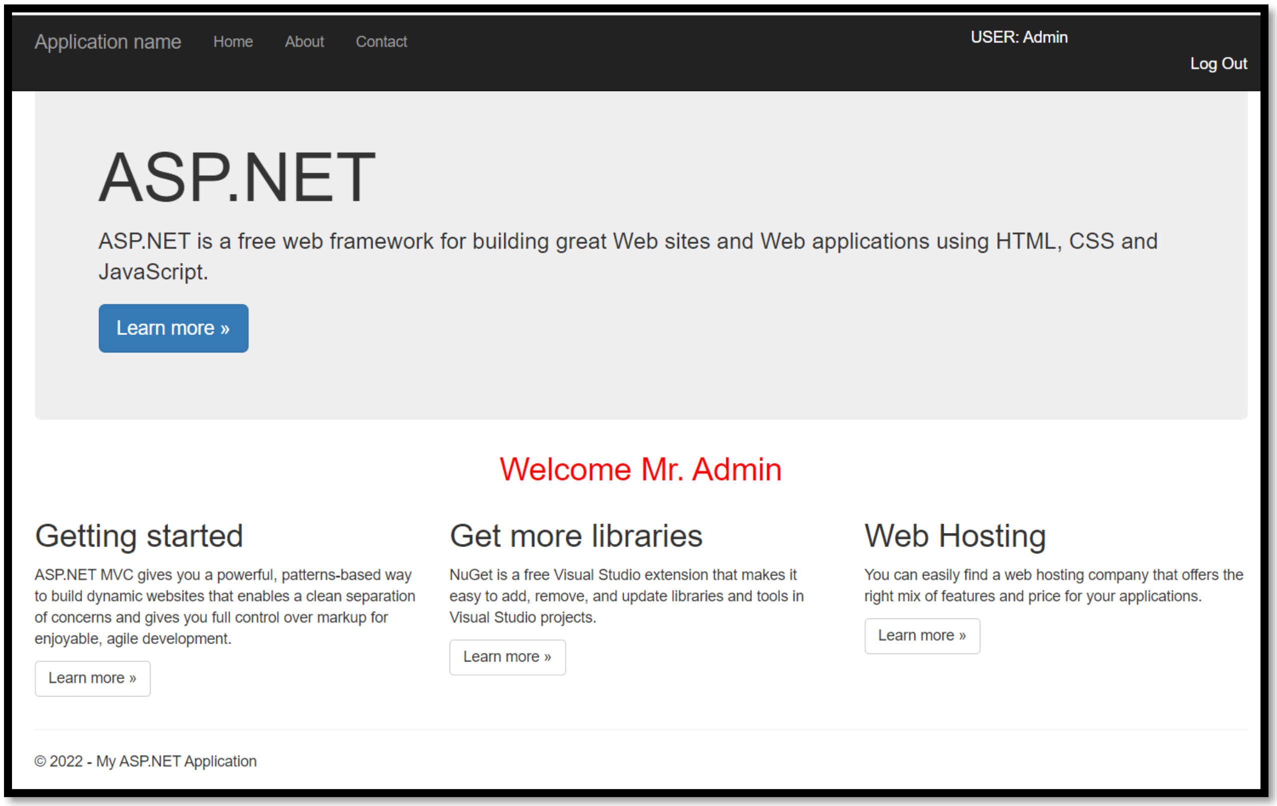
Login Page

Manual Setup Code : IFSG22LOKUYWIMSBGN4TI

872507

© 2022 - My ASP.NET Application

Step 20: If your code is correct, you will be **redirected** to your **index page** after clicking the submit button in the above screen.



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Welcome Mr. Admin

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

© 2022 - My ASP.NET Application

2. Time Boxing

Activity Name	Activity Time	Total Time
Login Systems + Setting up Visual studio Environment	3 mints + 5 mints	8 mints
Walk through Theory & Tasks	60 mints	60 mints
Implement Tasks	80 mints	80 mints
Evaluation Time	30 mints	30 mints
	Total Duration	178 mints

3. Objectives

After completing this lab the student should be able to:

a. *Understand the significance of Two-Factor Authentication (2FA) in enhancing security measures within ASP.NET MVC applications.*

b. *Implement 2FA using Google Authenticator as a secondary authentication method to fortify user account security in ASP.NET MVC applications.*

4. Lab Tasks/Practical Work

1. Develop a Two-Factor Authentication (2FA) Micro service using Google Authenticator in ASP.NET MVC for User Account Security Enhancement.