```
@page "/Account/Manage/EnableAuthenticator"

@using System.ComponentModel.DataAnnotations
@using System.Globalization
@using System.Text
@using System.Text.Encodings.Web
@using Microsoft.AspNetCore.Identity
@using BlazorApp2.Data
@using QRCoder
@using System.Drawing.Imaging
@using System.Drawing

@inject UserManager<ApplicationUser> UserManager
@inject IdentityUserAccessor UserAccessor
@inject UrlEncoder UrlEncoder
@inject IdentityRedirectManager RedirectManager
@inject ILogger<EnableAuthenticator> Logger

<PageTitle>Configure authenticator app</PageTitle>

@if (recoveryCodes is not null)
{
    <ShowRecoveryCodes RecoveryCodes="recoveryCodes.ToArray()"
StatusMessage="@message" />
}
else
{
    <StatusMessage Message="@message" />
    <h3>Configure authenticator app</h3>
    <div>
        <p>To use an authenticator app go through the following steps:</p>
        <ol class="list">
            <li>
                <p>
                    Download a two-factor authenticator app like Microsoft
Authenticator for
                    <a
href="https://go.microsoft.com/fwlink/?Linkid=825072">Android</a> and
                    <a href="https://go.microsoft.com/fwlink/?Linkid=825073">iOS</a>
or
                    Google Authenticator for
                    <a
href="https://play.google.com/store/apps/details?id=com.google.android.apps.authenti
cator2&amp;hl=en">Android</a> and
                    <a href="https://itunes.apple.com/us/app/google-
authenticator/id388497605?mt=8">iOS</a>.
                </p>
            </li>
            <li>
                <p>Scan the QR Code or enter this key <kbd>@sharedKey</kbd> into
your two factor authenticator app. Spaces and casing do not matter.</p>
                <div class="alert alert-info">Learn how to <a
href="https://go.microsoft.com/fwlink/?Linkid=852423">enable QR code
generation</a>.</div>
                <div><img alt="" src="@QRByte" width="300" class="mb-5" /></div>
                <div data-url="@QRByte"></div>
            </li>
            <li>
```

/*DIWAKAR DEVANGAM*/

```
                <p>
                    Once you have scanned the QR code or input the key above, your
two factor authentication app will provide you
                    with a unique code. Enter the code in the confirmation box
below.
                </p>
                <div class="row">
                    <div class="col-md-6">
                        <EditForm Model="Input" FormName="send-code"
OnValidSubmit="OnValidSubmitAsync" method="post">
                            <DataAnnotationsValidator />
                            <div class="form-floating mb-3">
                                <InputText @bind-Value="Input.Code" class="form-
control" autocomplete="off" placeholder="Please enter the code." />
                                <label for="code" class="control-label form-
label">Verification Code</label>
                                <ValidationMessage For="() => Input.Code"
class="text-danger" />
                            </div>
                            <button type="submit" class="w-100 btn btn-lg btn-
primary">Verify</button>
                            <ValidationSummary class="text-danger" role="alert" />
                        </EditForm>
                    </div>
                </div>
            </li>
        </ol>
    </div>
}

@code {
    private const string AuthenticatorUriFormat =
"otpauth://totp/{0}:{1}?secret={2}&issuer={0}&digits=6";

    private string? message;
    private ApplicationUser user = default!;
    private string? sharedKey;
    private string? authenticatorUri;
    private IEnumerable<string>? recoveryCodes;
    public string QRByte = "";
    [CascadingParameter]
    private HttpContext HttpContext { get; set; } = default!;

    [SupplyParameterFromForm]
    private InputModel Input { get; set; } = new();

    protected override async Task OnInitializedAsync()
    {
        user = await UserAccessor.GetRequiredUserAsync(HttpContext);
        await LoadSharedKeyAndQrCodeUriAsync(user);
    }
    public void GenerateQRCode(string QRUrl)
    {
        if (!string.IsNullOrEmpty(QRUrl))
        {
            using MemoryStream ms = new();
            QRCodeGenerator qrCodeGenerate = new();
```

```
            QRCodeData qrCodeData = qrCodeGenerate.CreateQrCode(QRUrl,
QRCodeGenerator.ECCLevel.Q);
            QRCode qrCode = new(qrCodeData);
            using Bitmap qrBitMap = qrCode.GetGraphic(20);
            qrBitMap.Save(ms, ImageFormat.Png);
            string base64 = Convert.ToBase64String(ms.ToArray());
            QRByte = string.Format("{0}", base64);
        }
    }
    private async Task OnValidSubmitAsync()
    {
        // Strip spaces and hyphens
        var verificationCode = Input.Code.Replace(" ", string.Empty).Replace("-",
string.Empty);

        var is2faTokenValid = await UserManager.VerifyTwoFactorTokenAsync(
            user, UserManager.Options.Tokens.AuthenticatorTokenProvider,
verificationCode);

        if (!is2faTokenValid)
        {
            message = "Error: Verification code is invalid.";
            return;
        }

        await UserManager.SetTwoFactorEnabledAsync(user, true);
        var userId = await UserManager.GetUserIdAsync(user);
        Logger.LogInformation("User with ID '{UserId}' has enabled 2FA with an
authenticator app.", userId);

        message = "Your authenticator app has been verified.";

        if (await UserManager.CountRecoveryCodesAsync(user) == 0)
        {
            recoveryCodes = await
UserManager.GenerateNewTwoFactorRecoveryCodesAsync(user, 10);
        }
        else
        {

RedirectManager.RedirectToWithStatus("Account/Manage/TwoFactorAuthentication",
message, HttpContext);
        }
    }

    private async ValueTask LoadSharedKeyAndQrCodeUriAsync(ApplicationUser user)
    {
        // Load the authenticator key & QR code URI to display on the form
        var unformattedKey = await UserManager.GetAuthenticatorKeyAsync(user);
        if (string.IsNullOrEmpty(unformattedKey))
        {
            await UserManager.ResetAuthenticatorKeyAsync(user);
            unformattedKey = await UserManager.GetAuthenticatorKeyAsync(user);
        }

        sharedKey = FormatKey(unformattedKey!);

        var email = await UserManager.GetEmailAsync(user);
```

```
            authenticatorUri = GenerateQrCodeUri(email!, unformattedKey!);
            GenerateQRCode(authenticatorUri);
        }

        private string FormatKey(string unformattedKey)
        {
            var result = new StringBuilder();
            int currentPosition = 0;
            while (currentPosition + 4 < unformattedKey.Length)
            {
                result.Append(unformattedKey.AsSpan(currentPosition, 4)).Append(' ');
                currentPosition += 4;
            }
            if (currentPosition < unformattedKey.Length)
            {
                result.Append(unformattedKey.AsSpan(currentPosition));
            }

            return result.ToString().ToLowerInvariant();
        }

        private string GenerateQrCodeUri(string email, string unformattedKey)
        {
            return string.Format(
                CultureInfo.InvariantCulture,
                AuthenticatorUriFormat,
                UrlEncoder.Encode("Microsoft.AspNetCore.Identity.UI"),
                UrlEncoder.Encode(email),
                unformattedKey);
        }

        private sealed class InputModel
        {
            [Required]
            [StringLength(7, ErrorMessage = "The {0} must be at least {2} and at max {1}
characters long.", MinimumLength = 6)]
            [DataType(DataType.Text)]
            [Display(Name = "Verification Code")]
            public string Code { get; set; } = "";
        }
}
```