Bahria University,

Karachi Campus

A picture containing text, emblem, symbol, crown

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_12\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Sign up page |
| 2 | Sign in page with google authenticator |

Submitted On:

20-06-2023

(Date: DD/MM/YY)

**LAB 12**

**Task 1 : Sign up page**

private readonly UserManager<IdentityUser> \_userManager;

private readonly SignInManager<IdentityUser> \_signInManager;

public AccountController(UserManager<IdentityUser> userManager, SignInManager<IdentityUser> signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

[HttpGet]

public IActionResult SignUp()

{

return View();

}

[HttpPost]

public async Task<IActionResult> SignUp(SignUpViewModel model)

{

if (ModelState.IsValid)

{

var user = new IdentityUser

{

UserName = model.Email,

Email = model.Email,

PhoneNumber = model.PhoneNumber

};

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

// Generate the email confirmation token

var emailConfirmationToken = await \_userManager.GenerateEmailConfirmationTokenAsync(user);

var emailConfirmationLink = Url.Action("ConfirmEmail", "Account",

new { userId = user.Id, token = emailConfirmationToken }, Request.Scheme);

// Send the email confirmation link to the user

// ...

// Redirect to the 2FA setup page

return RedirectToAction("SetupTwoFactorAuthentication", new { userId = user.Id });

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError("", error.Description);

}

}

}

return View(model);

}

[HttpGet]

public async Task<IActionResult> SetupTwoFactorAuthentication(string userId)

{

var user = await \_userManager.FindByIdAsync(userId);

if (user == null)

{

return NotFound();

}

var providers = await \_userManager.GetValidTwoFactorProvidersAsync(user);

var model = new SetupTwoFactorViewModel

{

UserId = userId,

AvailableProviders = providers.ToList()

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> SetupTwoFactorAuthentication(SetupTwoFactorViewModel model)

{

var user = await \_userManager.FindByIdAsync(model.UserId);

if (user == null)

{

return NotFound();

}

var provider = model.SelectedProvider;

var token = await \_userManager.GenerateTwoFactorTokenAsync(user, provider);

var result = await \_userManager.SetTwoFactorEnabledAsync(user, true);

if (!result.Succeeded)

{

// Handle the error

return RedirectToAction("Error");

}

if (provider == "Email")

{

var emailConfirmationLink = Url.Action("ConfirmEmail", "Account",

new { userId = user.Id, token = token }, Request.Scheme);

}

else if (provider == "Phone")

{}

return RedirectToAction("VerifyTwoFactorAuthentication", new { userId = user.Id, provider });

}

[HttpGet]

public IActionResult VerifyTwoFactorAuthentication(string userId, string provider)

{

var model = new VerifyTwoFactorViewModel

{

UserId = userId,

Provider = provider};

return View(model);}

[HttpPost]

public async Task<IActionResult> VerifyTwoFactorAuthentication(VerifyTwoFactorViewModel model)

{

var user = await \_userManager.FindByIdAsync(model.UserId);

if (user == null)

{

return NotFound();

}

var result = await \_signInManager.TwoFactorSignInAsync(model.Provider, model.Code, false, false);

if (result.Succeeded)

{

// Redirect to the successful sign-in page

return RedirectToAction("SignedIn");

}

else

{

ModelState.AddModelError("", "Invalid verification code.");

return View(model);

}

}

A screenshot of a computer

Description automatically generated with low confidenceA screenshot of a login box

Description automatically generated with medium confidence

**Task#2: Sign in page with google authenticator.**

using Google.Authenticator;

using GoogleAuthenticatorDemo.Models;

using Newtonsoft.Json.Linq;

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;

using System.Web.UI.WebControls;

using System.Xml.Linq;

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);

if (login.UserName == "Admin" && login.Password == "25897")

{

Session["UserName"] = login.UserName;

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();

}

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");

}

}

}

Login.cs code

@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 >



