Practice Project

**Use Selenium, NUnit, and SpecFlow to Test a UI Prototype of Bank Webapp**

LoginController.cs

using Microsoft.AspNetCore.Mvc;

namespace BankWebApp.Controllers

{

public class LoginController : Controller

{

public IActionResult Index()

{

return View();

}

[HttpGet]

public IActionResult Login()

{

return View();

}

[HttpPost]

public IActionResult Login(string username, string password)

{

return RedirectToAction("Index", "Dashboard", new { username = username });

}

}

}

Login.cshtml

@{

ViewData["Title"] = "Login";

}

<h2>Login</h2>

<**form** method="post" **asp-controller**="Login" **asp-action**="Login">

<label>Username:</label>

<input type="text" name="username" />

<br />

<label>Password:</label>

<input type="password" name="password" />

<br />

<button type="submit">Login</button>

</**form**>

DashBoardController.cs

using Microsoft.AspNetCore.Mvc;

namespace BankWebApp.Controllers

{

public class DashboardController : Controller

{

public IActionResult Index(string username)

{

ViewBag.Username = username;

return View();

}

[HttpPost]

public IActionResult Logout()

{

// Add logic for logout

// Redirect to Login page

return RedirectToAction("Login", "Login");

}

}

}

Index.cshtml

@{

ViewData["Title"] = "Dashboard";

}

<h2>Welcome, @ViewBag.Username!</h2>

<p>Bank Dashboard</p>

<**form** method="post" **asp-controller**="Dashboard" **asp-action**="Logout">

<button type="submit">Logout</button>

</**form**>

<div class="mt-3">

<h4>Account Actions</h4>

<button class="btn btn-success" onclick="window.location.href='/Dashboard/Deposit'">Deposit</button>

<button class="btn btn-warning" onclick="window.location.href='/Dashboard/Withdraw'">Withdraw</button>

<button class="btn btn-success" onclick="window.location.href='/Dashboard/Transfer'">Transfer</button>

<!-- Add more buttons for other account actions as needed -->

</div>

Dockerfile

#See https://aka.ms/customizecontainer to learn how to customize your debug container and how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base

WORKDIR /app

EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

ARG BUILD\_CONFIGURATION=Release

WORKDIR /src

COPY ["BankWebApp/BankWebApp.csproj", "BankWebApp/"]

RUN dotnet restore "./BankWebApp/./BankWebApp.csproj"

COPY . .

WORKDIR "/src/BankWebApp"

RUN dotnet build "./BankWebApp.csproj" -c $BUILD\_CONFIGURATION -o /app/build

FROM build AS publish

ARG BUILD\_CONFIGURATION=Release

RUN dotnet publish "./BankWebApp.csproj" -c $BUILD\_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "BankWebApp.dll"]