# use SplashKit to build a web server

Based on the functions in SplashKit, we can build our own web server.

In this report, we will follow the structure below to learn how to build a web server using SplashKit:

1. How to start the server
2. How to respond to a request
3. How to implement routing on the server

## How to start server

After initializing the program by **skm dotnet new console**, use **SplashKit.StartWebServer()** to start a web server.

|  |
| --- |
| WebServer server = SplashKit.StartWebServer(); |

addtion, use **SplashKit.StopWebServer()** to stop server

|  |
| --- |
| SplashKit.StopWebServer(server); |

## How to respond to a request

When the server receives a request from Chrome or any client, it must respond to the request; otherwise, Chrome or the client will continue to wait for a response, potentially resulting in a long delay.

use **SplashKit.SendResponse()** to send a text response, use **SplashKit.SendHtmlFileResponse()** to send an html file as a response.

|  |
| --- |
| SplashKit.SendResponse(request, "add user successfuly"); |
| SplashKit.SendHtmlFileResponse(request, "contact.html"); |

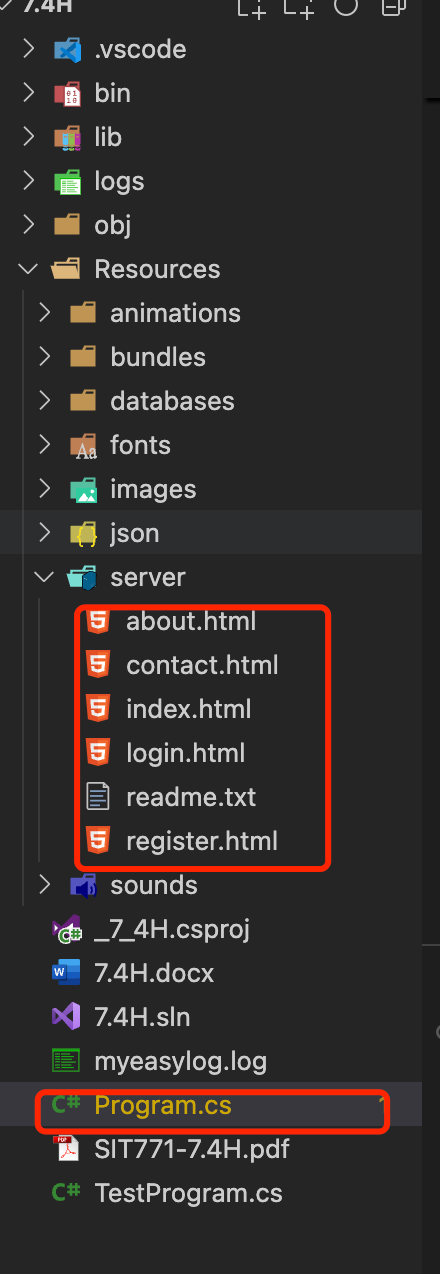
## How to implement routing on the server

There are many types of requests in the HTTP protocol, including GET, POST, PUT, DELETE, and OPTIONS. In this task, we will use GET and POST, which are the most commonly used types of HTTP requests.

use **SplashKit.IsGetRequestFor()** and **SplashKit.IsPostRequestFor()** to do the request check.

# Task Demonstration

## program structure

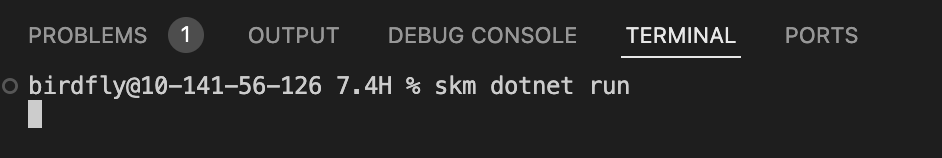


/server: this is the directory to store all the html resources;

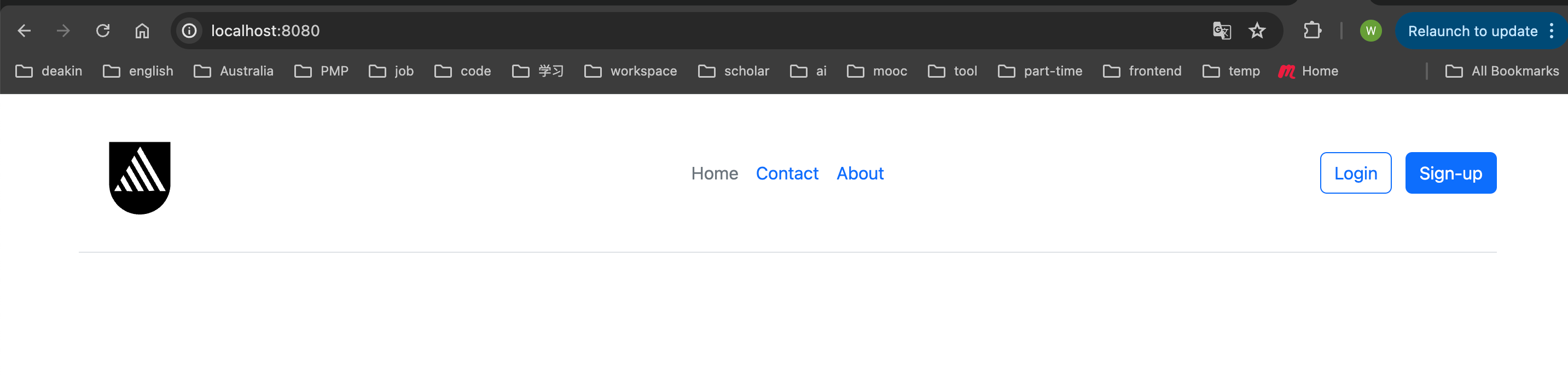
Program.cs: all the logic of server.

## Running Display

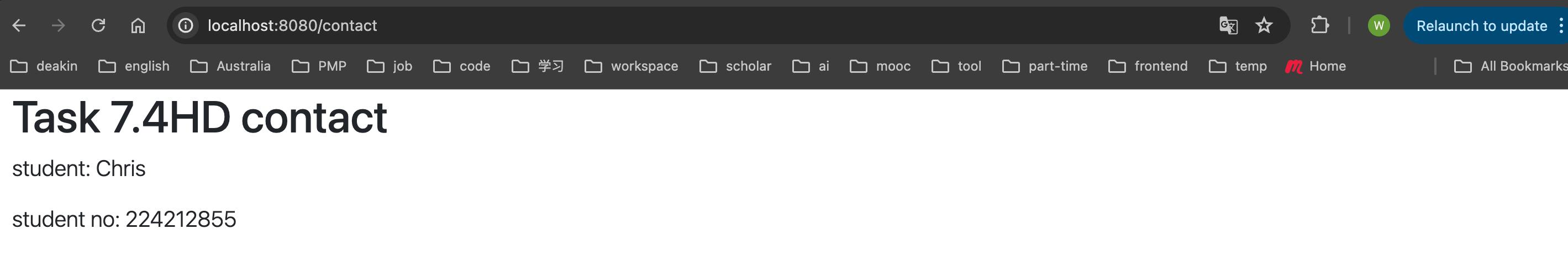
### Start server



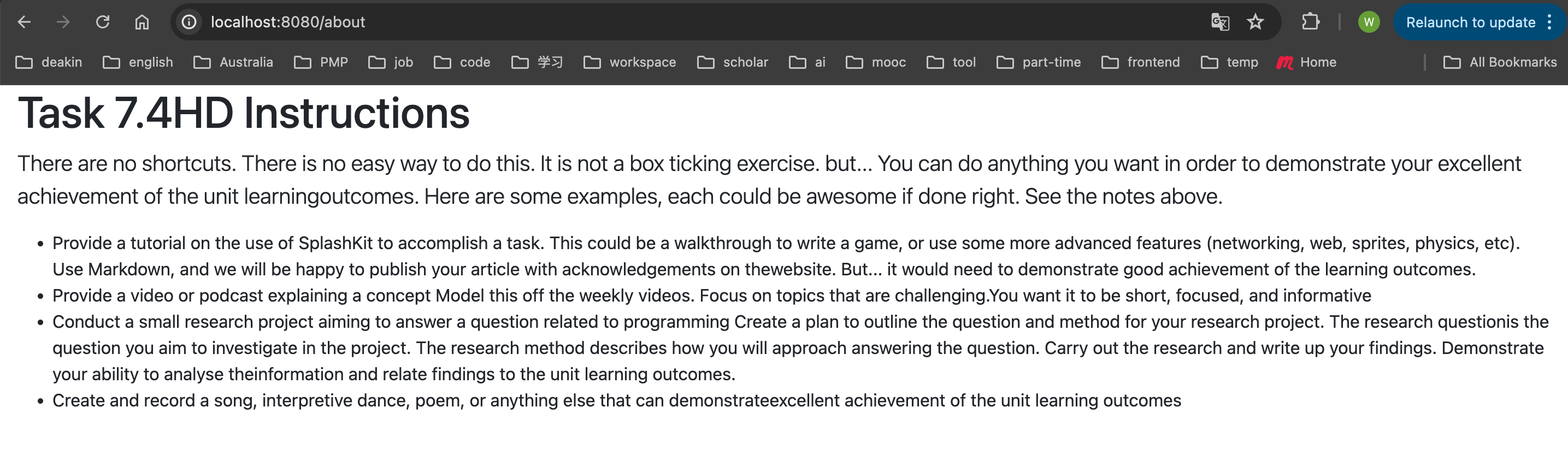
### Get Request / or /index



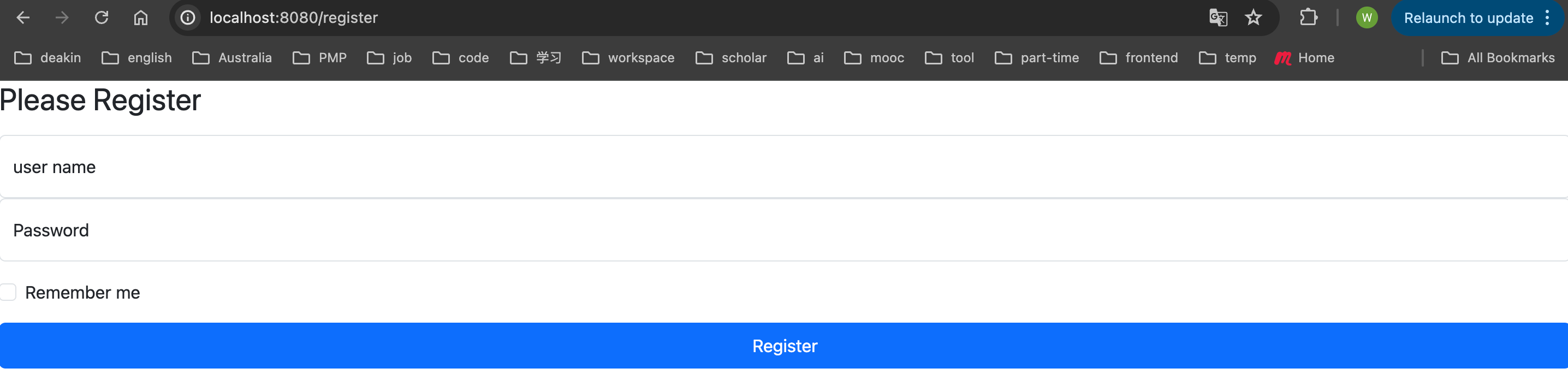
### Get Request /Contact



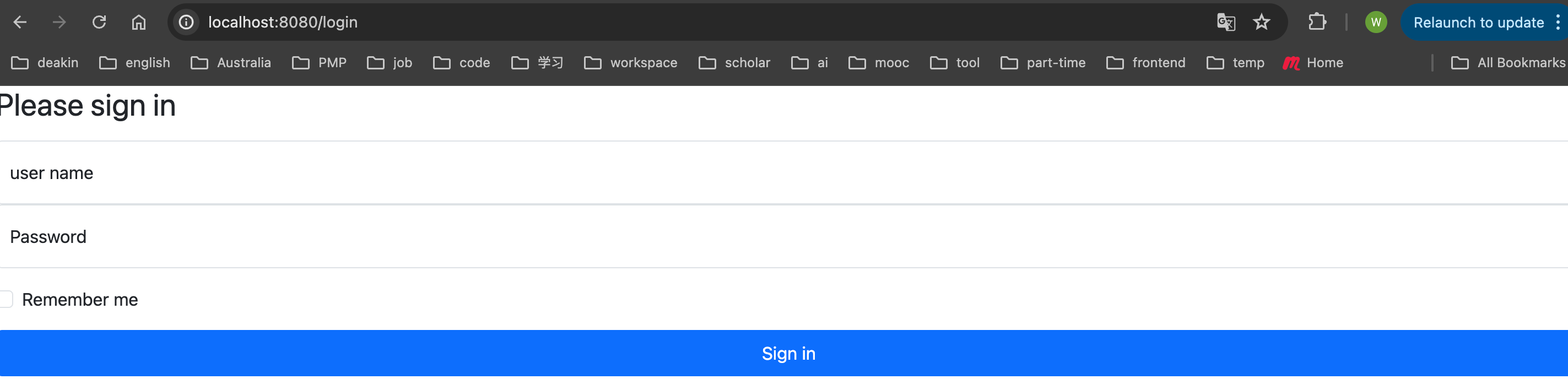
### Get Request /about



### Get Request /register



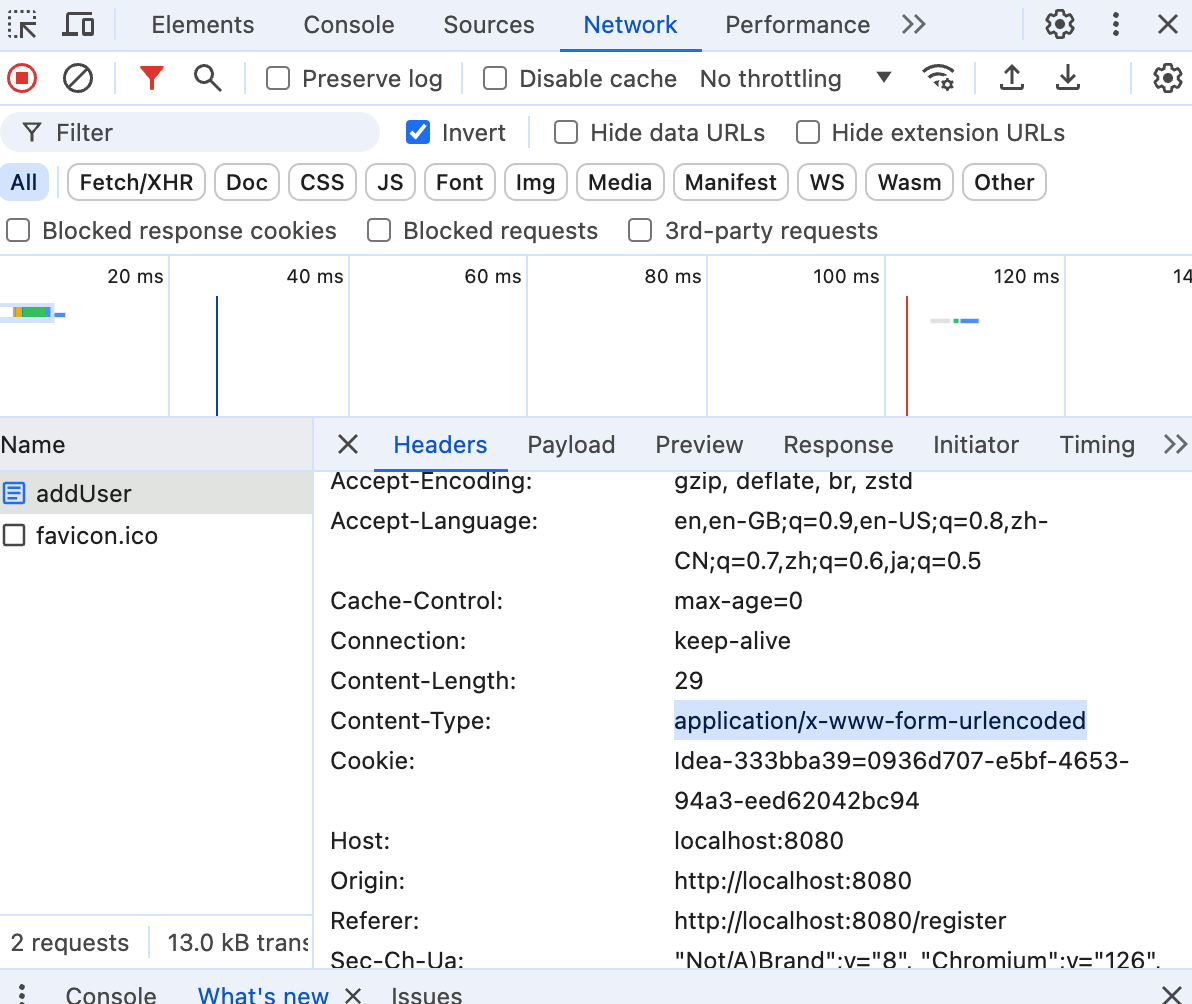
### Get Request /login

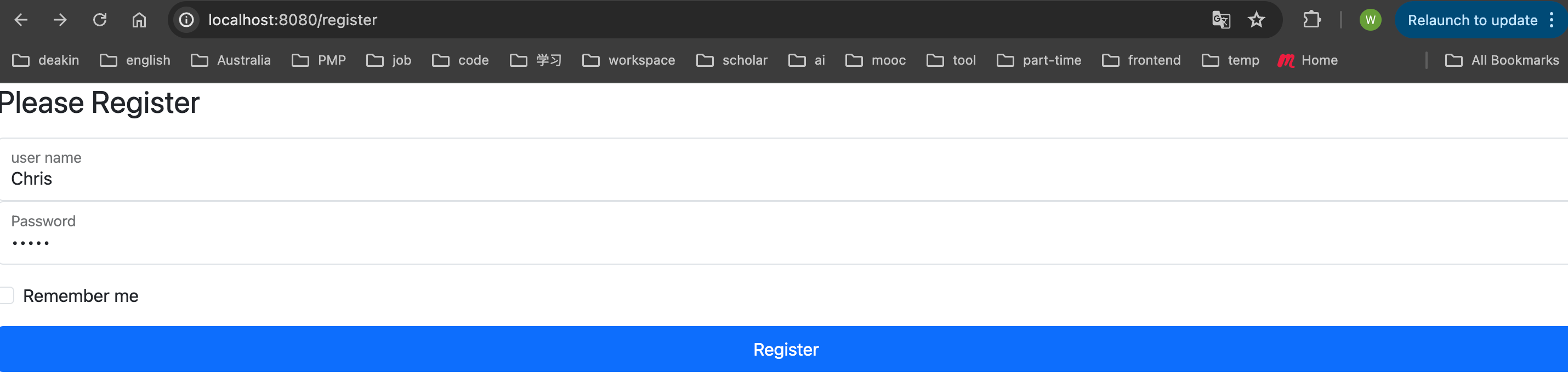


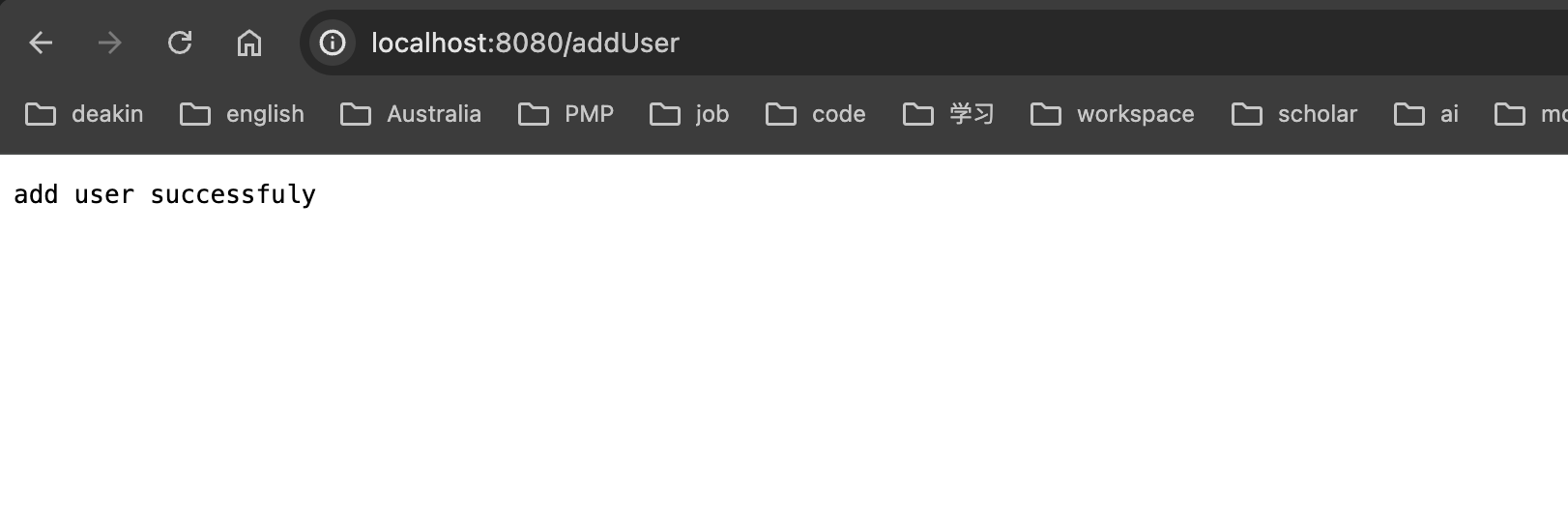
### Post Request /addUser

To submit a POST request using a form that contains **username=Chris** and **password=chris**, when we don't specify an **enctype**, the form will submit the content using the default **application/x-www-form-urlencoded** encoding, when the form is submitted, the data is sent in the format:

**username=Chris&password=chris**.

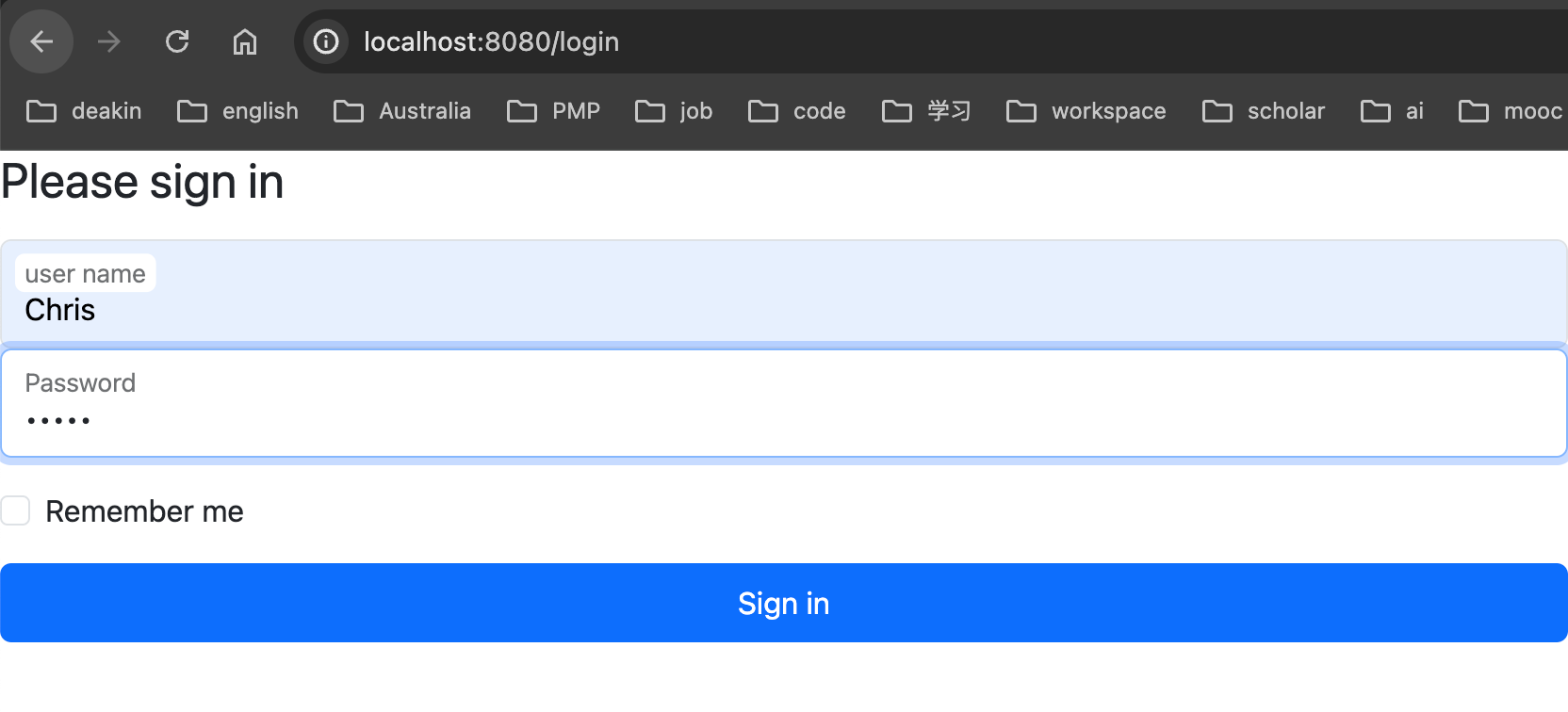




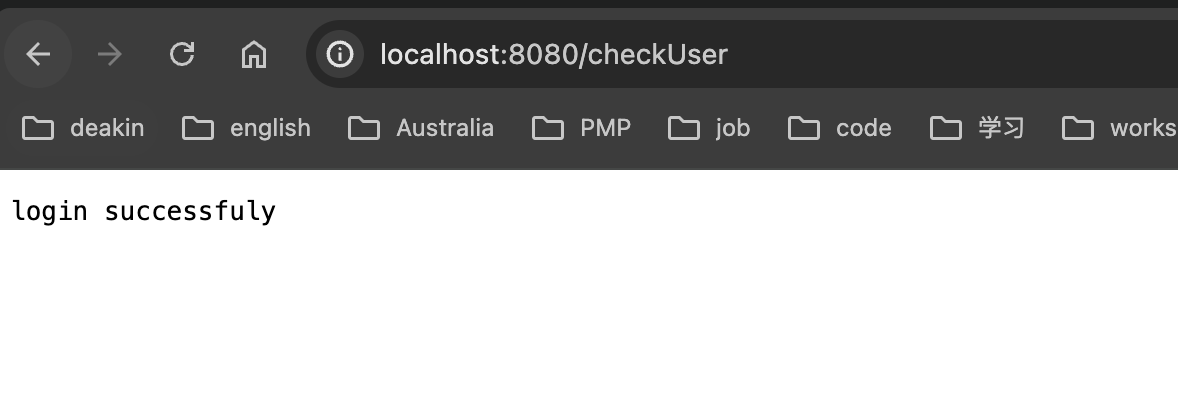


### Post Request /checkUser

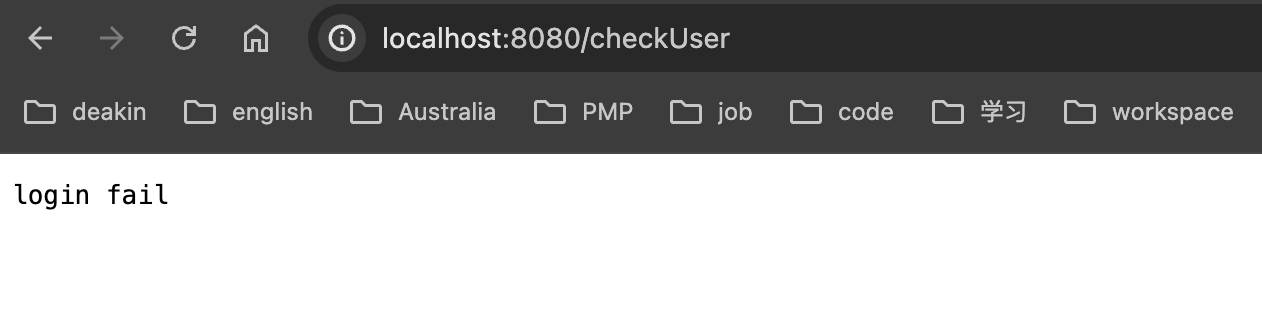
In this program, we use a Dictionary to store user information. If a user has not registered, they will not be able to log in. The Dictionary acts as a simple user database, where the username and password are stored as key-value pairs.



Login success



Login fail



# Sources Code

Program.cs

|  |
| --- |
| using System;  using System.Text.Json.Nodes;  using SplashKitSDK;  namespace \_7\_4H  {  public class Program  {  private static Dictionary<string, string> \_userMap = new Dictionary<string, string>();  public static void Main()  {  WebServer server = SplashKit.StartWebServer();  HttpRequest request;  request = SplashKit.NextWebRequest(server);  while (!SplashKit.IsGetRequestFor(request, "/quit"))  {  SplashKit.WriteLine("I got a request for " + SplashKit.RequestURI(request));  if (SplashKit.IsGetRequestFor(request, "/login") || SplashKit.IsGetRequestFor(request, "/login.html"))  {  SplashKit.SendHtmlFileResponse(request, "login.html");  }  else if (SplashKit.IsPostRequestFor(request, "/checkUser"))  {  string userInfo = request.Body;  if (userInfo != null && userInfo != "")  {  string[] userArr = userInfo.Split("&");  if (userArr != null && userArr.Length > 1)  {  string username = (userArr[0].Split("="))[1];  string password = (userArr[1].Split("="))[1];  bool flag = \_userMap.ContainsKey(username);  SplashKit.WriteLine(\_userMap.GetValueOrDefault(username));  if (flag && \_userMap.GetValueOrDefault(username) == password)  {  SplashKit.SendResponse(request, "login successfuly");  }  }  }  SplashKit.SendResponse(request, "login fail");  }  else if (SplashKit.IsGetRequestFor(request, "/register") || SplashKit.IsGetRequestFor(request, "/register.html"))  {  SplashKit.SendHtmlFileResponse(request, "register.html");  }  else if (SplashKit.IsPostRequestFor(request, "/addUser"))  {  string userInfo = request.Body;  if (userInfo != null && userInfo != "")  {  string[] userArr = userInfo.Split("&");  if (userArr != null && userArr.Length > 1)  {  string username = (userArr[0].Split("="))[1];  string password = (userArr[1].Split("="))[1];  \_userMap.TryAdd(username, password);  SplashKit.SendResponse(request, "add user successfuly");  }  }  SplashKit.SendResponse(request, "add user fail");  }  else if (SplashKit.IsGetRequestFor(request, "/contact") || SplashKit.IsGetRequestFor(request, "/contact.html"))  {  SplashKit.SendHtmlFileResponse(request, "contact.html");  }  else if (SplashKit.IsGetRequestFor(request, "/about") || SplashKit.IsGetRequestFor(request, "/about.html"))  {  SplashKit.SendHtmlFileResponse(request, "about.html");  }  else  {  SplashKit.SendHtmlFileResponse(request, "index.html");  }  SplashKit.WriteLine("Waiting for a request - navigate to http://localhost:8080");  SplashKit.WriteLine("To end - navigate to http://localhost:8080/quit");  // Get the next request that the server has  request = SplashKit.NextWebRequest(server);  }  SplashKit.StopWebServer(server);  }  }  } |

# Relevant Materials

SplashKit:

<https://splashkit.io/guides/networking/0-getting-started-with-servers/>

Bootstrap:

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

C#:

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/program-structure/>