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.

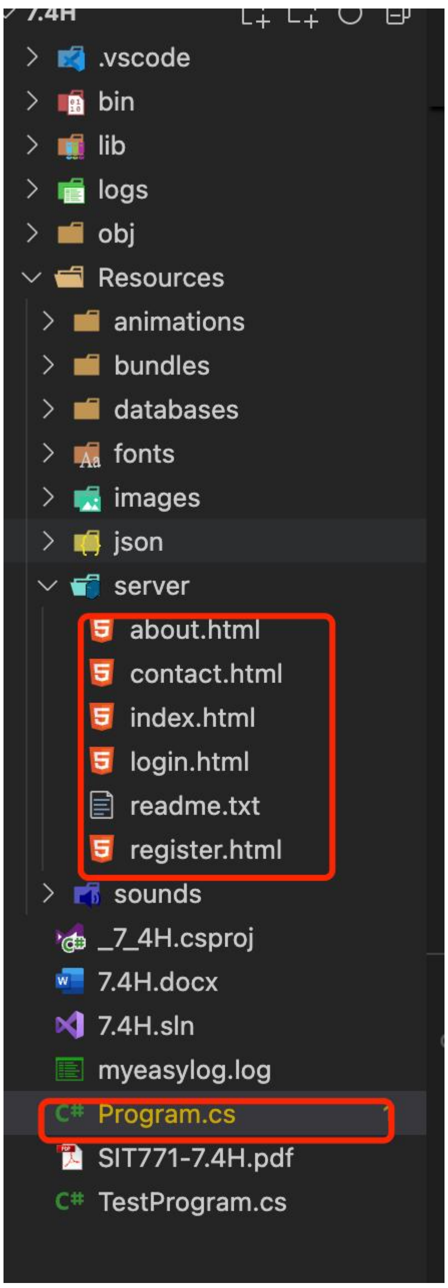
```
SplashKit.SendResponse(request, "add user successfuly");
SplashKit.SendHtmlFileResponse(request, "contact.html");
```

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



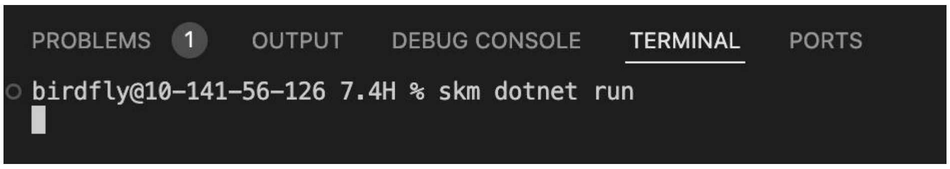
```
7.4H
> .vscode
> bin
> lib
> logs
> obj
v Resources
  > animations
  > bundles
  > databases
  > fonts
  > images
  > json
  v server
      about.html
      contact.html
      index.html
      login.html
      readme.txt
      register.html
  > sounds
  _7_4H.csproj
  7.4H.docx
  7.4H.sln
  myeasylog.log
  Program.cs
  SIT771-7.4H.pdf
  TestProgram.cs
```

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

Program.cs: all the logic of server.

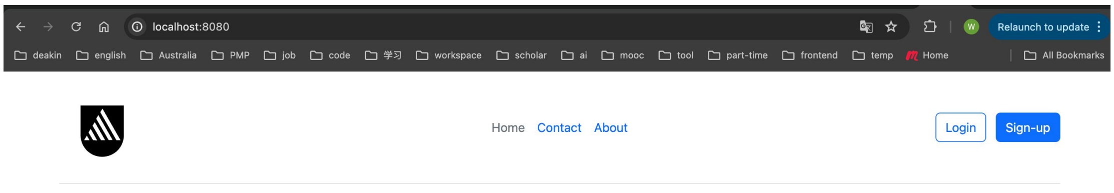
Running Display

Start server

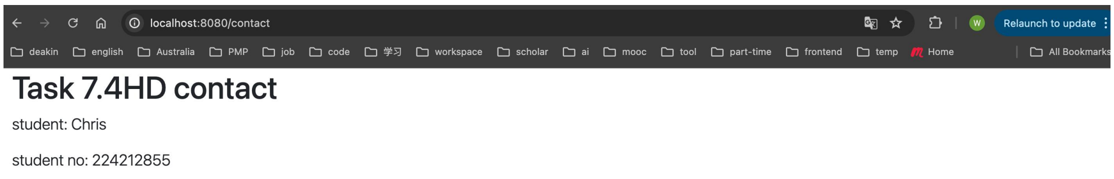


```
birdfly@10-141-56-126 7.4H % skm dotnet run
```

Get Request / or /index



Get Request /Contact

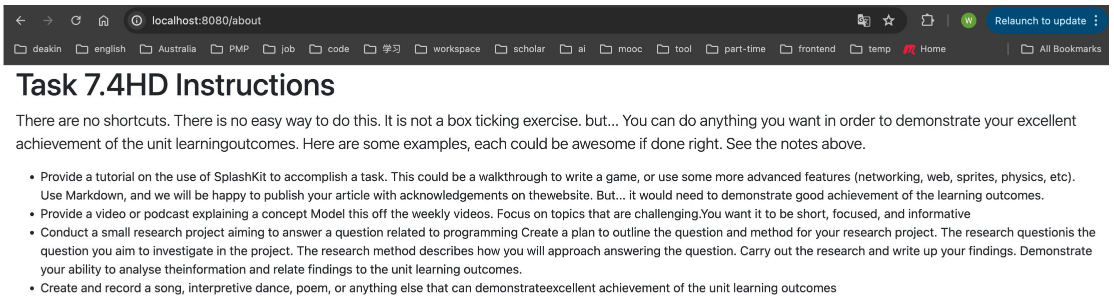


Task 7.4HD contact

student: Chris

student no: 224212855

Get Request /about

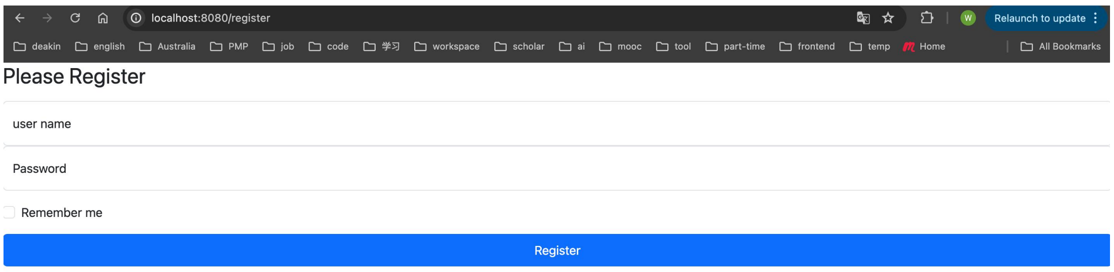


Task 7.4HD Instructions

There are no shortcuts. There is no easy way to do this. It is not a box ticking exercise. but... You can do anything you want in order to demonstrate your excellent achievement of the unit learningoutcomes. Here are some examples, each could be awesome if done right. See the notes above.

- Provide a tutorial on the use of SplashKit to accomplish a task. This could be a walkthrough to write a game, or use some more advanced features (networking, web, sprites, physics, etc). Use Markdown, and we will be happy to publish your article with acknowledgements on thewebsite. But... it would need to demonstrate good achievement of the learning outcomes.
- Provide a video or podcast explaining a concept Model this off the weekly videos. Focus on topics that are challenging.You want it to be short, focused, and informative
- Conduct a small research project aiming to answer a question related to programming Create a plan to outline the question and method for your research project. The research questionis the question you aim to investigate in the project. The research method describes how you will approach answering the question. Carry out the research and write up your findings. Demonstrate your ability to analyse theinformation and relate findings to the unit learning outcomes.
- Create and record a song, interpretive dance, poem, or anything else that can demonstrateexcellent achievement of the unit learning outcomes

Get Request /register


localhost:8080/register

Please Register

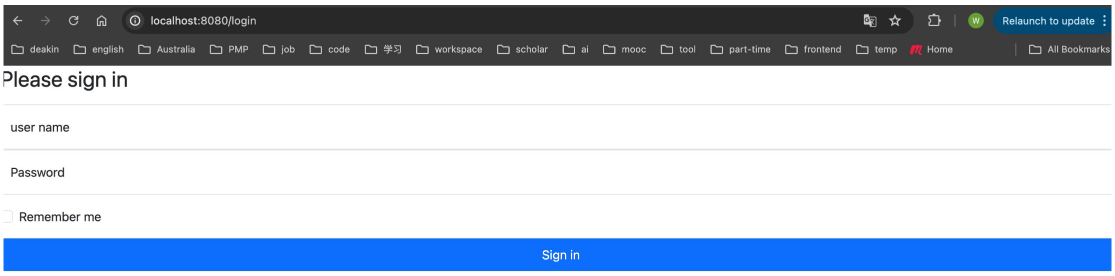
user name

Password

Remember me

Register

Get Request /login


localhost:8080/login

Please sign in

user name

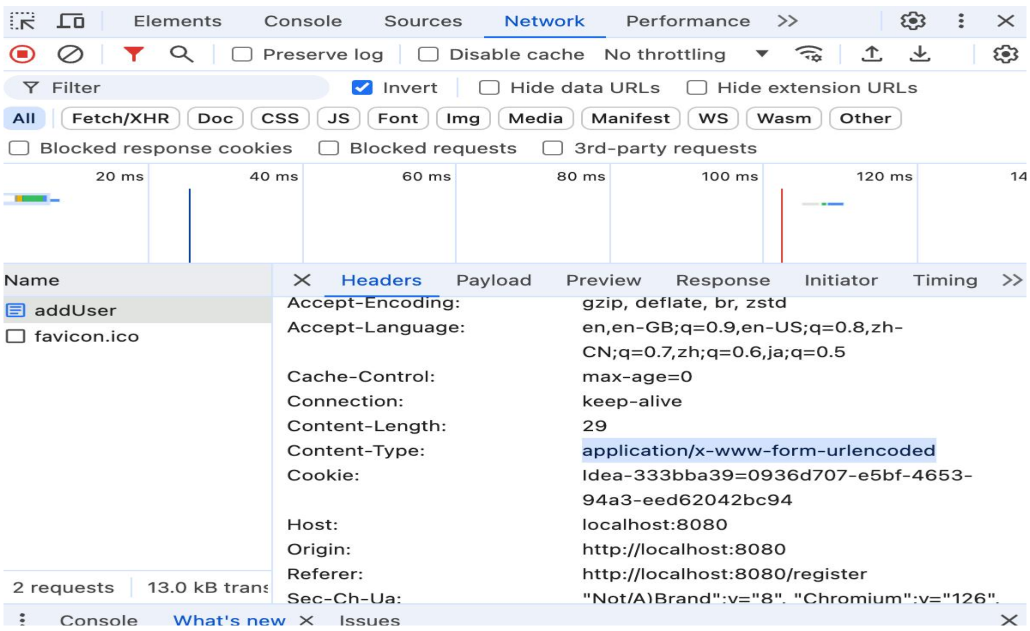
Password

Remember me

Sign in

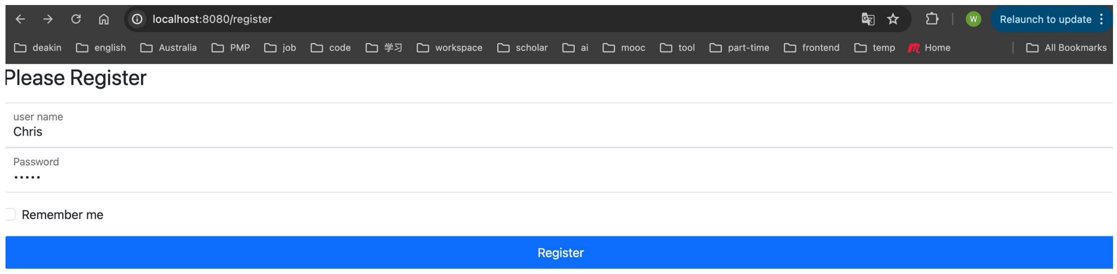
Post Request /addUser

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.



addUser
favicon.ico

```
Accept-Encoding:      gzip, deflate, br, zstd
Accept-Language:      en,en-GB;q=0.9,en-US;q=0.8,zh-
                      CN;q=0.7,zh;q=0.6,ja;q=0.5
Cache-Control:        max-age=0
Connection:           keep-alive
Content-Length:       29
Content-Type:         application/x-www-form-urlencoded
Cookie:               Idea-333bba39=0936d707-e5bf-4653-
                      94a3-eed62042bc94
Host:                 localhost:8080
Origin:               http://localhost:8080
Referer:              http://localhost:8080/register
Sec-Ch-Ua:            "Not/A)Brand";v="8", "Chromium";v="126",
```

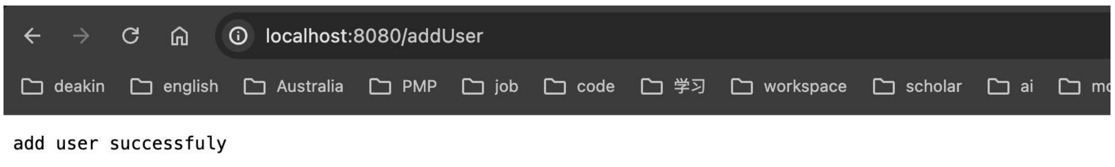


Please Register

user name
Chris

Password
•••••

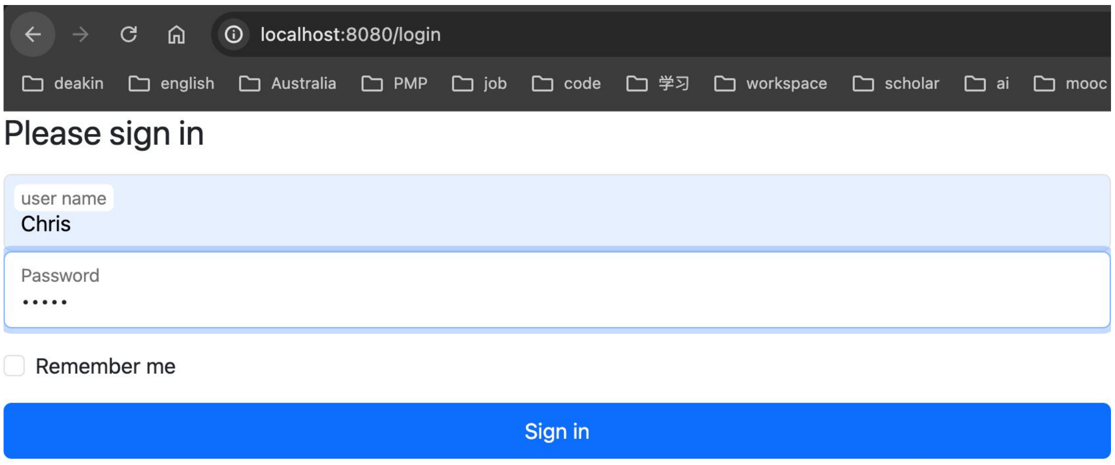
Remember me



```
add user successfuly
```

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.



localhost:8080/login

Please sign in

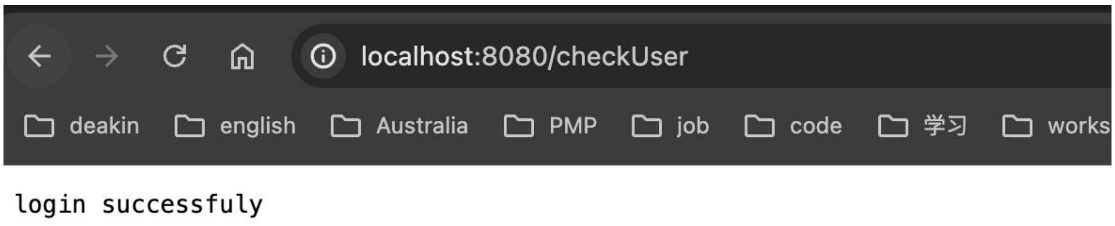
user name
Chris

Password
•••••

Remember me

Sign in

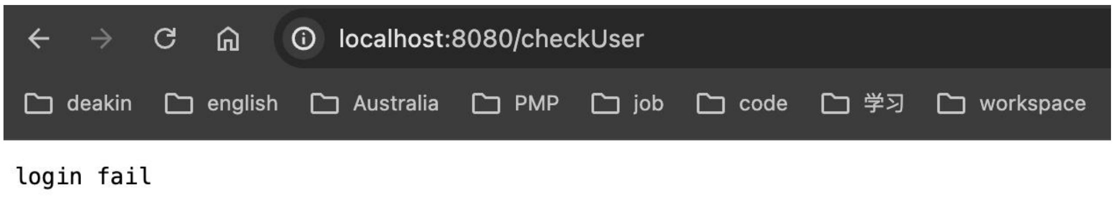
Login success



localhost:8080/checkUser

```
login successfuly
```

Login fail



localhost:8080/checkUser

```
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/