 Statement and Confirmation of Own Work

|  |
| --- |
| ***A signed copy of this form must be submitted with every assignment.***  ***If the statement is missing your work may not be marked.*** |

Student Declaration

I confirm the following details:

|  |  |
| --- | --- |
| Candidate Name: | AMAAN AL MIR |
| Candidate ID Number: |  |
| Qualification: | L5DC |
| Unit: | Computing Project – Project Report |
| Centre: | APTECH QATAR |
| I have read and understood both NCC Education’s *Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.  I confirm that this is my own work and that I have not colluded or plagiarised any part of it. | |
| Candidate Signature: |  |
| Date: | Feb 6, 2023 |



**OPS020\_dec16\_Candidate+Statement+of+Own+Work.doc**

# Cover Page

**WEB CHAT**

PROJECT REPORT

Prepared by

**Amaan Al Mir**

# Table of Contents

Get Started

# Get Started

WebChat is designed for private groups where they can spend time together. A place where they can talk every day and hang out more often.

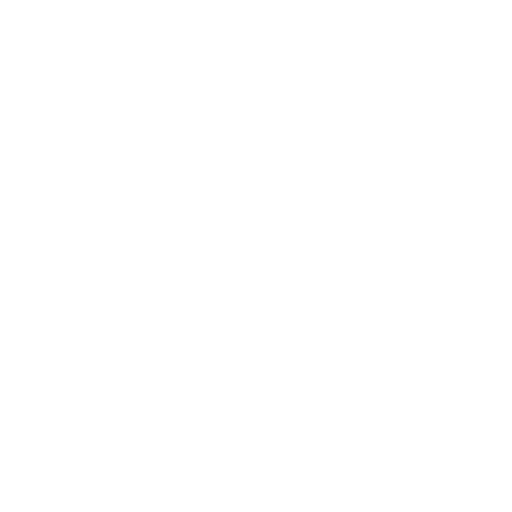
This web application is designed to be responsive and user-friendly to let individuals and groups communicate easily. The design has very simple layers of textures and functions. Unlike other online social media and communication services, this application wasn’t designed for commercial purposes, although business may also use it for their private purposes.

This documentation contains guides and reference materials to help you learn, integrate, optimize and troubleshoot your use of WebChat application.

(Discord, n.d.)

Contents

# Developer Guide

****

**< DEV />**

**GUIDE**



## Aims & Objectives

These are the main goals when changing the contents of this web application.

* Security of the accounts and their data.
* Ease of use of application.
* Website responsiveness across all devices and browsers.
* Smooth behavior of controls, settings, and functions.
* Error handling, both client and server side, across all devices, browsers and other supported platforms.
* Less data consumption.
* Better performance and benchmark.
* Privacy of user data and their activities.
* User management done on web as much as possible.
* Backup and restore point of everything.

(ASP.NET, n.d.)

(Web Development with ASP.NET MVC and Core, 2022)

(Final Report Structure In Computing Project, 2023)

## Prerequisites

This web application is created using ASP.NET MVC 5. Here are the softwares, programs and languages used to create this application.

Softwares & Programs

**Installation files can be found in the Prerequisites folder.**

Programming Languages

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Other Libraries

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Version** | 3.6.0 | 5.2.1 | 5.15.4 |

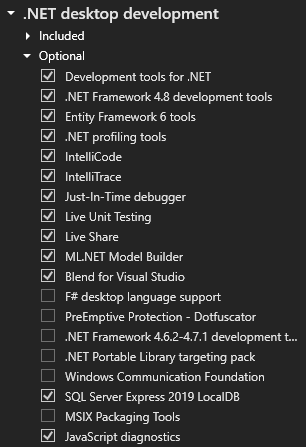
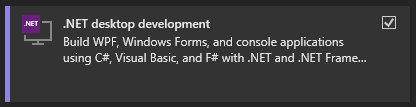
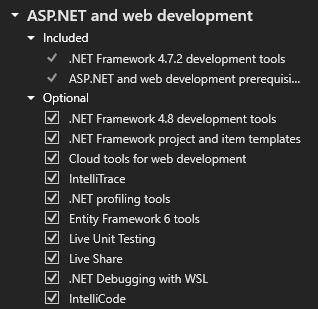
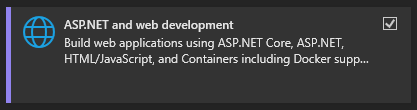
(ASP.NET, n.d.)

(SQL Server, 2022)

(Web Development with ASP.NET MVC and Core, 2022)

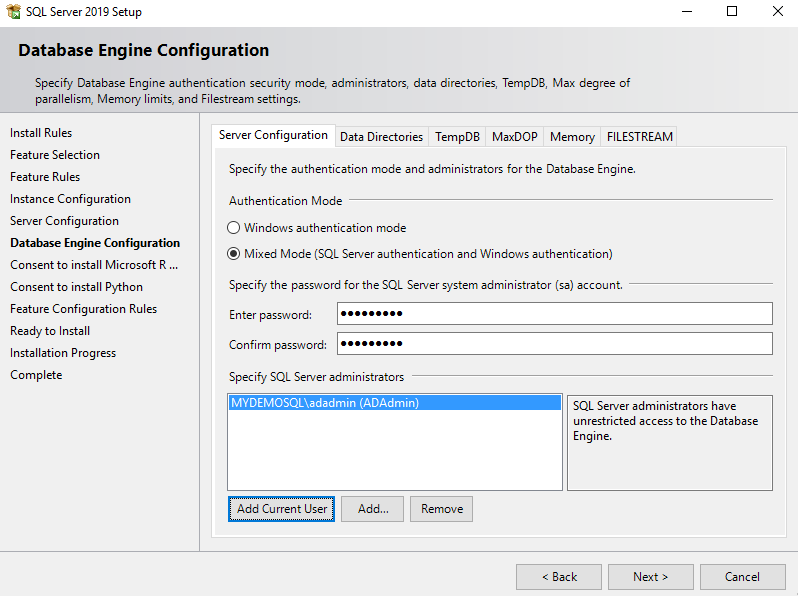
### Installing Visual Studio

When installing Visual Studio, make sure these options are checked.



### Installing SQL Server

When installing SQL Server, make sure to choose **Mixed Mode** for **Database Engine Configuration**.



## Integrations

SignalR

SendGrid

## Application Contents

Discover what’s included in WebChat.

**Note:** Only editable files are displayed on the tree below. Do not touch any other file you find inside the application that is not listed below.

WebChat \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| * App\_Start   + BundleConfig.cs   + IdentityConfig.cs   + Startup.Auth.cs * Content   + Site.js * Controllers   + AccountController.cs   + DirectMessagesController.cs   + FriendsController.cs   + HomeController.cs   + ManageController.cs   + ProfilesController.cs * Models   + DirectMessages.cs   + IdentityModel.cs   + Messages.cs   + Profiles.cs   + ViewModel.cs * Scripts   + Site.js * Chathub.cs * Startup.cs * Web.config | * Views   + Account     - \_ExternalLoginsListPartial.cshtml     - ConfirmEmail.cshtml     - ConfirmEmailInfo.cshtml     - ExternalLoginConfirmation.cshtml     - ExternalLoginFailure.cshtml     - ForgotPassword.cshtml     - ForgotPasswordConfirmation.cshtml     - Login.cshtml     - Register.cshtml     - ResetPassword.cshtml     - ResetPasswordConfirmation.cshtml     - SendCode.cshtml     - VerifyCode.cshtml   + DirectMessages     - Index.cshtml   + Friends     - Index.cshtml   + Home     - Index.cshtml   + Manage     - AddPhoneNumber.cshtml     - ChangePassword.cshtml     - Index.cshtml     - ManageLogins.cshtml     - SetPassword.cshtml     - VerifyPhoneNumber.cshtml   + Profiles   + Shared     - \_Layout.cshtml     - \_LoginPartial.cshtml     - Error.cshtml     - Lockout.cshtml |

## Database Contents

1. Users Table [dbo.AspNetUsers]

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** |  |
| Id | nvarchar(128) |  |
| Email | nvarchar(256) |  |
| EmailConfirmed | bit |  |
| PasswordHash | nvarchar(max) |  |
| SecurityStamp | nvarchar(max) |  |
| PhoneNumber | nvarchar(max) |  |
| PhoneNumberConfirmed | bit |  |
| TwoFactorEnabled | bit |  |
| LockoutEndDateUtc | datetime |  |
| LockoutEnabled | bit |  |
| AccessFailedCount | int |  |
| UserName | nvarchar(256) |  |

1. External Logins Table [dbo.AspNetUserLogins]

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** |  |
| LoginProvider | nvarchar(128) |  |
| ProviderKey | nvarchar(256) |  |
| UserId | bit |  |

1. User Profiles Table [dbo.Profiles]

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** |  |
| Email | nvarchar(256) |  |
| JoinDate | datetime |  |
| DisplayName | nvarchar(16) |  |
| Avatar | varbinary(max) |  |
| Bio | nvarchar(200) |  |
| Status | varchar(50) |  |
| Friends | nvarchar(max) |  |
| OutgoingFriendRequests | nvarchar(max) |  |
| IncomingFriendRequests | nvarchar(max) |  |

1. Lobby Messages Table [dbo.Messages]

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** |  |
| MsgID | int |  |
| Msg | nvarchar(max) |  |
| Img | varbinary(max) |  |
| Email | nvarchar(256) |  |
| DateTime | datetime |  |

1. Direct Messages Table [dbo.DirectMessages]

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** |  |
| Id | int |  |
| Sender | nvarchar(256) |  |
| Receiver | nvarchac(256) |  |
| Msg | nvarchar(max) |  |
| Img | varbinary(max) |  |
| DateTime | datetime |  |

## ER Diagram

|  |  |
| --- | --- |
| **AspNetUserLogins** | |
|  | LoginProvider |
|  | ProviderKey |
|  | UserId |

|  |  |
| --- | --- |
| **AspNetUsers** | |
|  | Id |
|  | Email |
|  | EmailConfirmed |
|  | PasswordHash |
|  | SecurityStamp |
|  | PhoneNumber |
|  | PhoneNumberConfirmed |
|  | TwoFactorEnabled |
|  | LockoutEndDateUtc |
|  | LockoutEnabled |
|  | AccessFailedCount |
|  | UserName |

|  |  |
| --- | --- |
| **Profiles** | |
|  | Email |
|  | JoinDate |
|  | DisplayName |
|  | Avatar |
|  | Bio |
|  | Status |
|  | Friends |
|  | OutgoingFriendRequests |
|  | IncomingFriendRequests |

|  |  |
| --- | --- |
| **Messages** | |
|  | MsgId |
|  | Msg |
|  | Img |
|  | Email |
|  | DateTime |

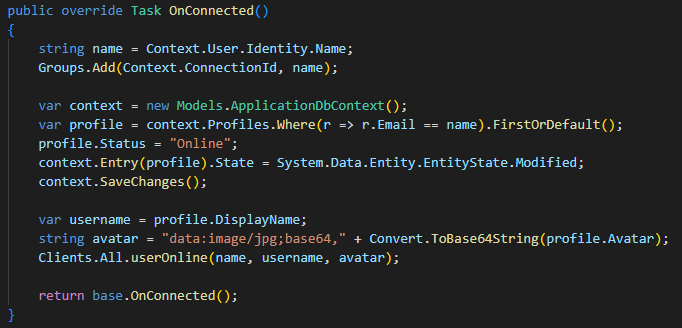
|  |  |
| --- | --- |
| **DirectMessages** | |
|  | Id |
|  | Sender |
|  | Receiver |
|  | Msg |
|  | Img |
|  | DateTime |

## Source Code

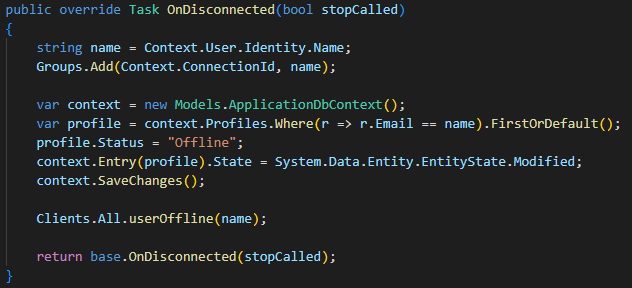
### ChatHub (SignalR)

**Initializing chat system (SignalR) in ChatHub.cs.**

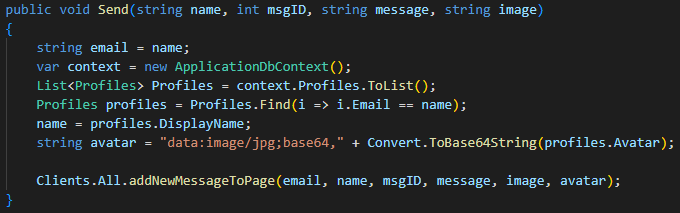
Setting user’s status to “online” when they connect to the application.



Setting user’s status to “offline” when they disconnect from the application.

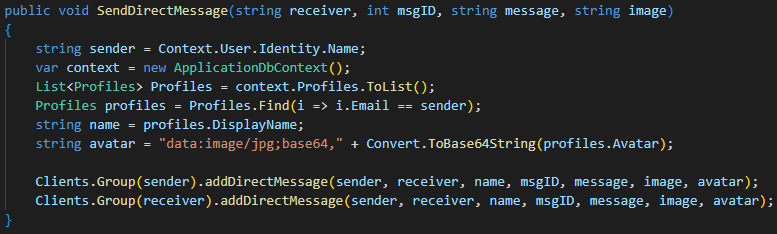


Setting up a function to send messages.



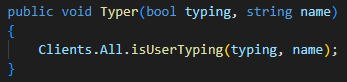
This function will return the sent message to all users.

Setting up Direct Message function.



This function will send a message to specific user only.

A function to display user is typing.

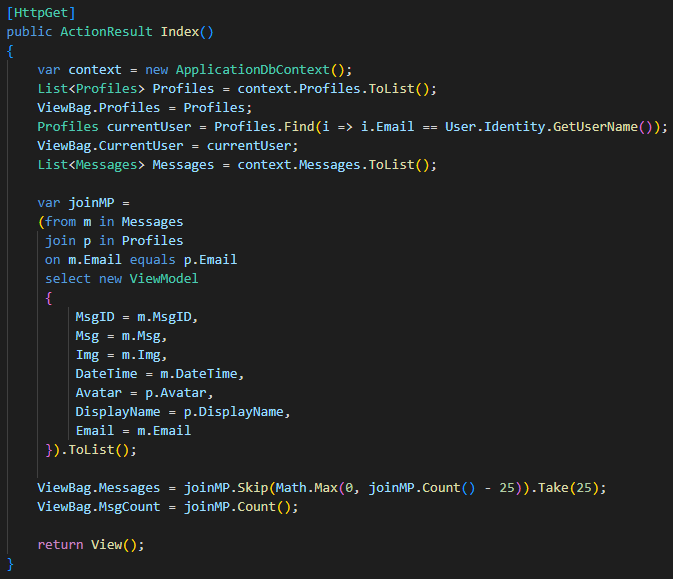


### Lobby

**Setting up the Lobby in Home/Index.cshtml.**

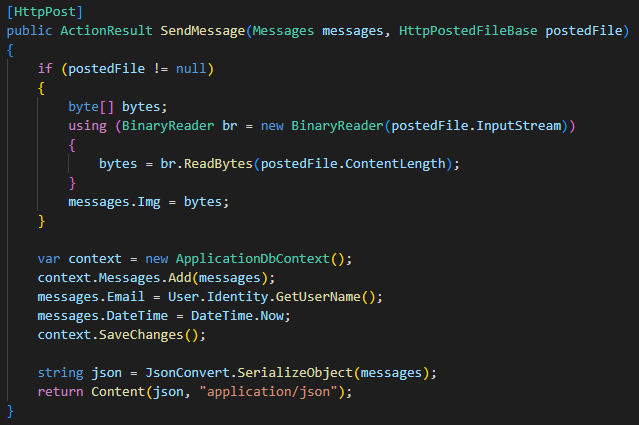
Only three models are used here; AspNetUsers, Profiles, and Messages. Most of the functions are performed client side using Ajax to prevent page refresh and for fast response.

Setting up Home Controller using C#.



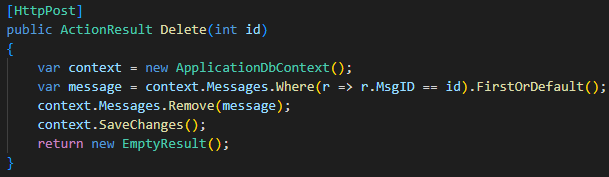
This function loads last 25 messages sent by users in the Lobby.

Defining a function to let users send messages in the Lobby.

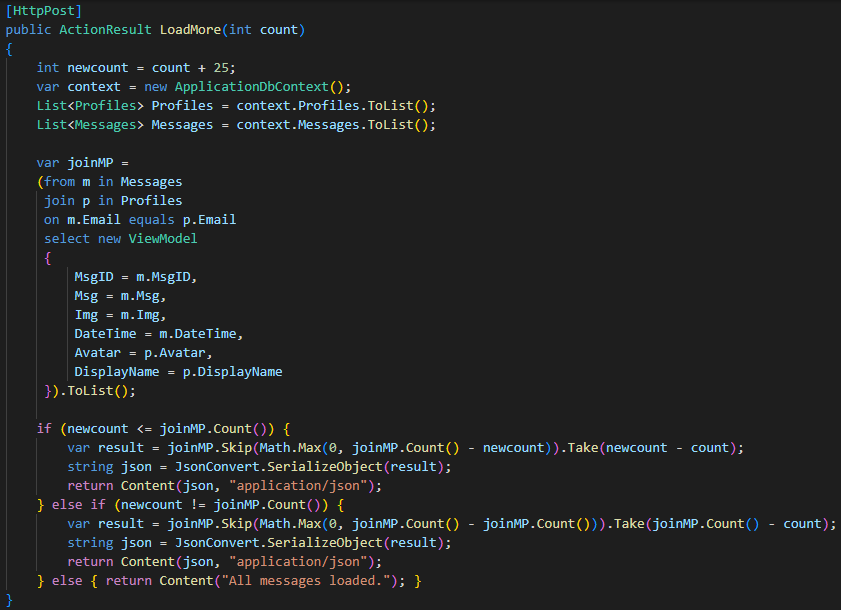


Here, we are posting a message from a form in Index.cshtml which will look soon, and converting the image, if given, to byte and saving in database.

Writing a function to let users delete a specified message that they posted.



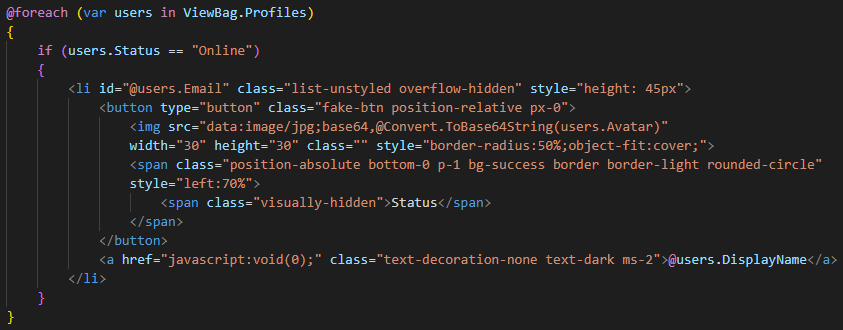
Since the app is only displaying 25 messages on start. Here’s a function to let users load more messages.



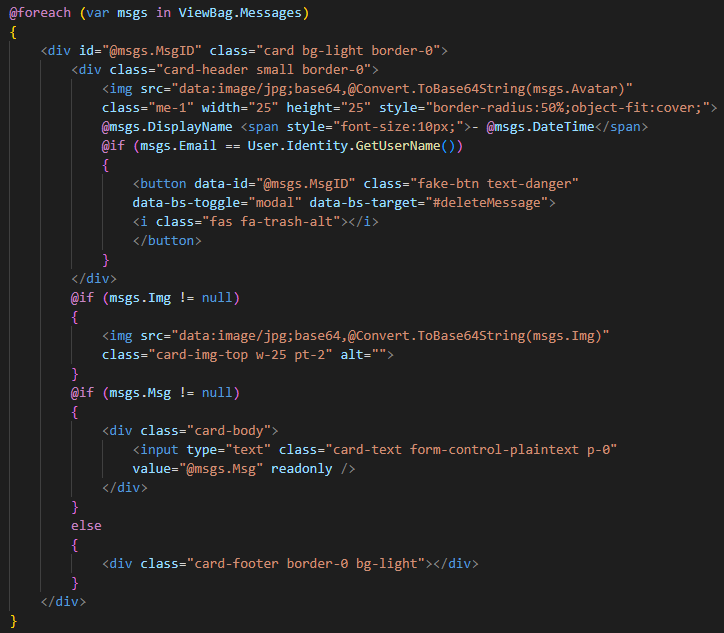
This will also return 25 messages.

**Setting up the views page using HTML and C#.**

Displaying a list of online users which is updated using JavaScript.

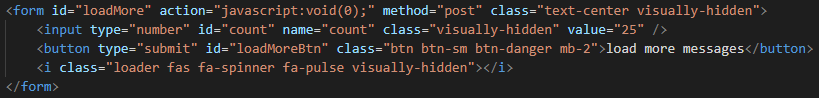


Loading messages which are stored in database when the page loads.



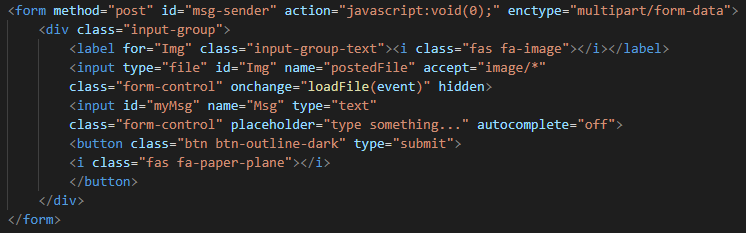
The above code returns the contents of Index Action from the Home Controller.

Since we are displaying only 25 messages when the page is loaded for the first time, we will add a button to load more messages.



This code returns the content of LoadMore Action defined in the Home Controller.

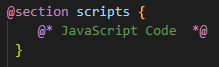
Adding a form to let users send new messages to the Lobby.



This HTML code is for posting new messages to the Home Controller which is handled by SendMessage Action.

Next up we have JavaScript where we will use ajax methods to request and submit data to or from the server.

Before we setup JavaScript, it is important to render script using this code given below at the bottom of the page to put functions below the shared file scripts.



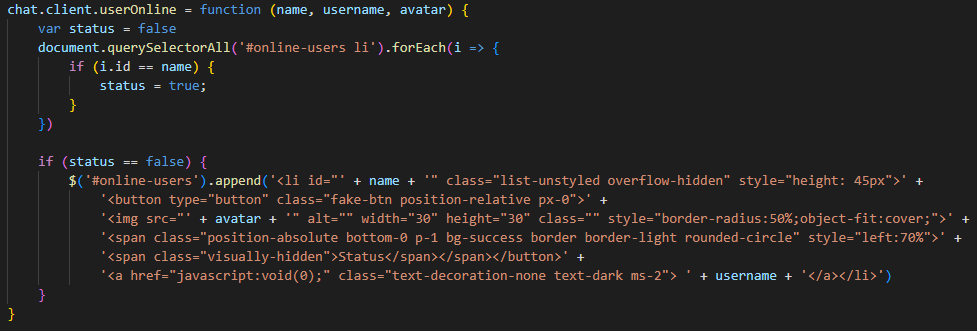
**Setting up JavaScript functions on Index.cshtml.**

Store the ChatHub connection to a variable.

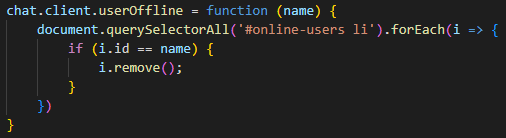


This is important as most of the functions are designed to work in real-time.

Updating the list of online users when a new user joins the Lobby.

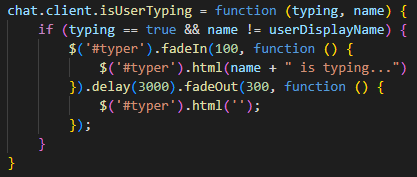


Updating the list of online users when a user leaves the Lobby.



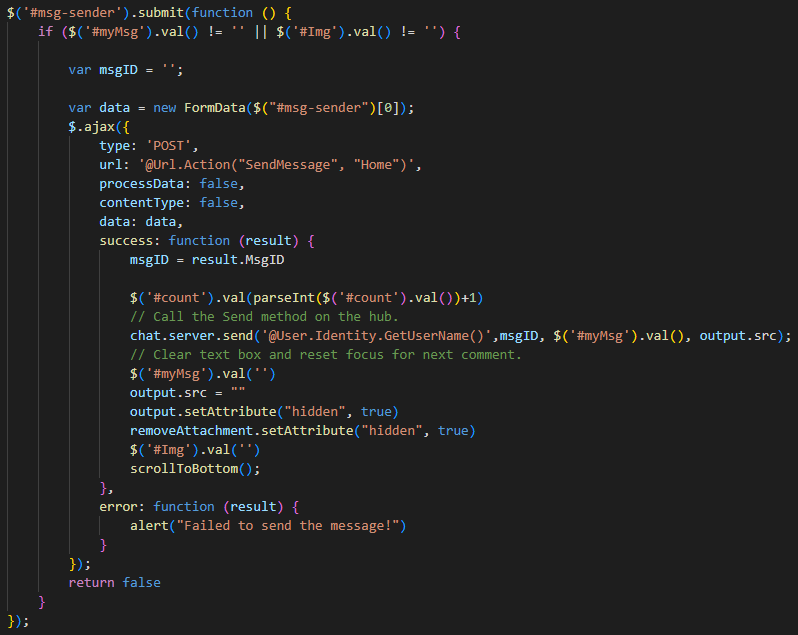
This will the remove the user’s DisplayName and Avatar from the list of online users.

Function to display user is typing in real-time.



This function is executed when a user presses a key. With the help of a little debounce function, which is given at the end of this section, it prevents from overloading the DOM. Let’s say user is constantly typing, so in order to prevent triggering the function on each key press, we are adding a simple timeout function.

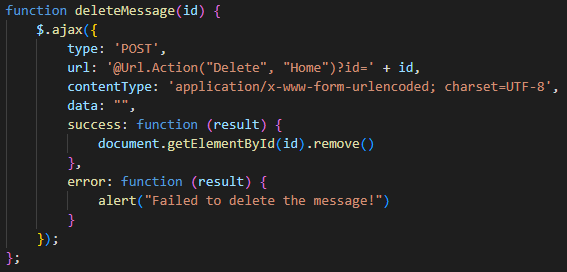
Function to send message.



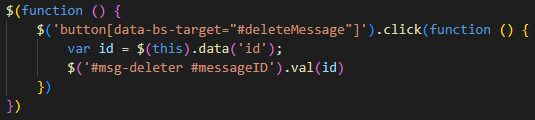
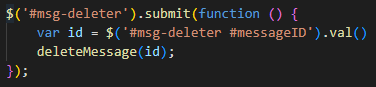
And displaying it to all users in real-time.



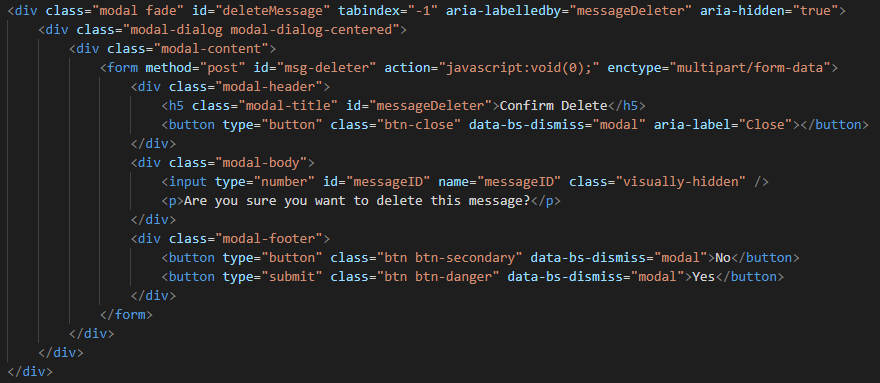
Function for deleting a message.



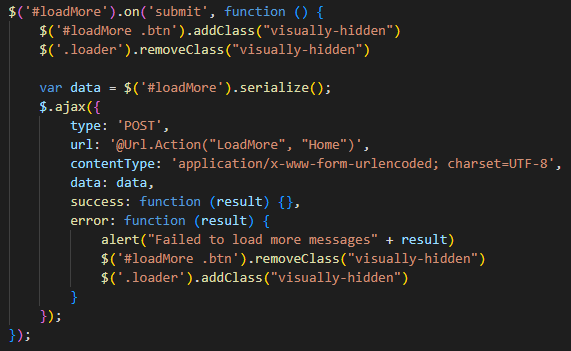
This function is called from Scripts/site.js.

This is connected to the message deletion confirmation modal.



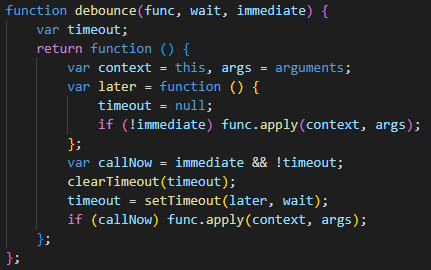
Function assigned to the button for loading more messages.



On success, it retrieves more 25 messages in a dictionary from the server which is handles by the Home Controller.

Due to the lengthy size of the code, it is not displayed here. But it is in the same format as when the page is loaded for the first time.

Here’s the debounce function we discussed earlier.



(Polesny, 2021)

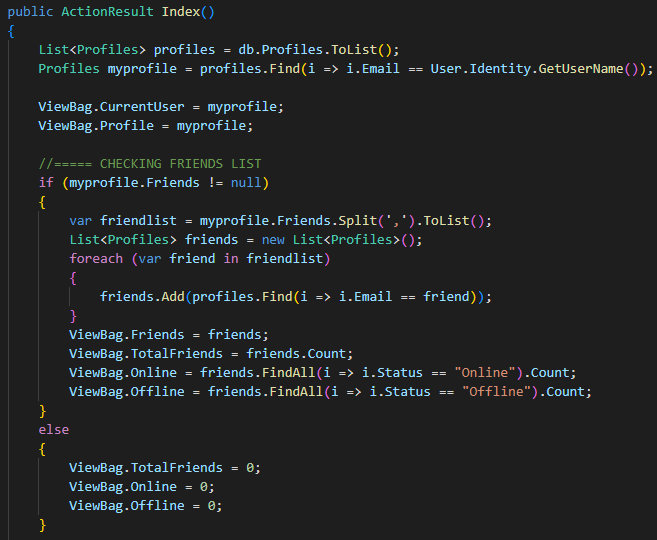
### Friends

**Setting up FriendsController.cs.**

Please note that due to lengthy text, codes have been modified and shortened for better understanding.

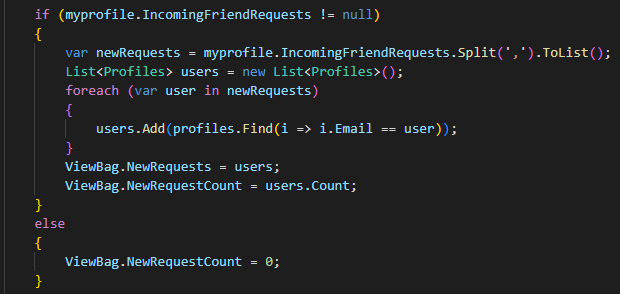
The is the starting of the first function which is for the default page, Index.cshtml.

This is to retrieve the list of all users who are friends with the current user.

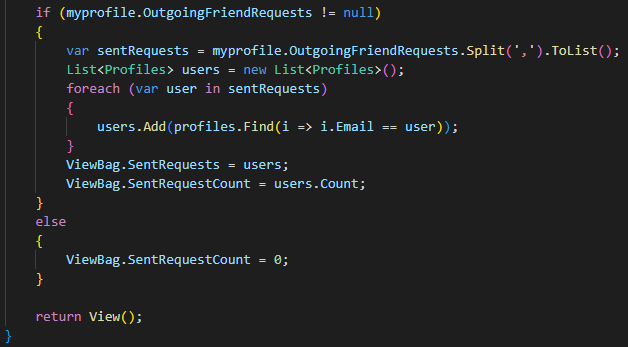


This function returns list of friends and number of friends to the Friends Index page.

Retrieving list of all friend requests sent by other users to current user.

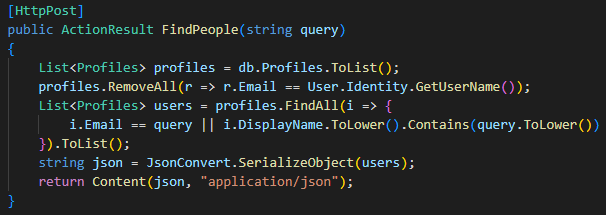


Retrieving list of friend requests sent to other users.



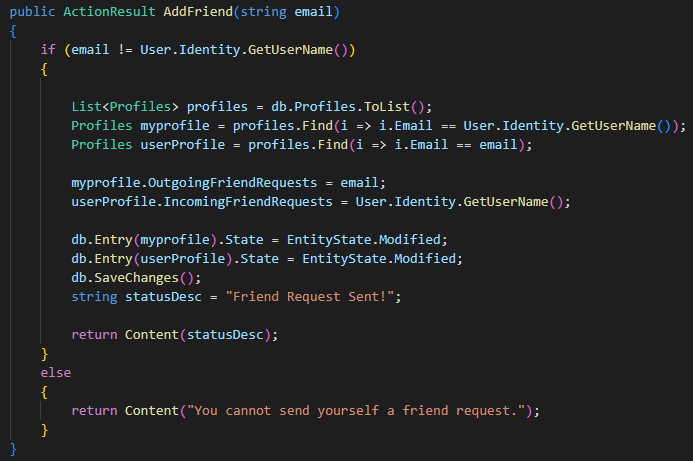
This was the end of first function, Index Action.

**Finding other users on the platform.**



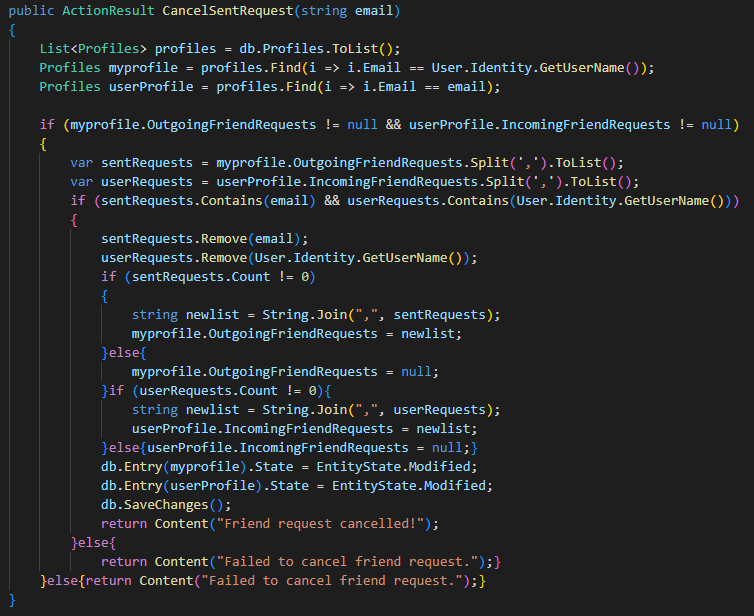
This function will search for users with given DisplayName or Email.

**Sending a friend request.**

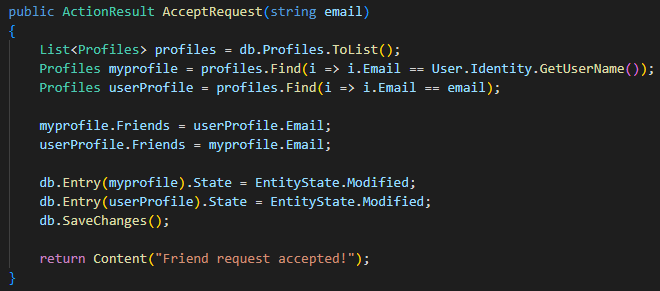


This function will add the other user’s email to current user’s OutgoingFriendRequests list and current to user’s email to other user’s IncomingFriendRequests list.

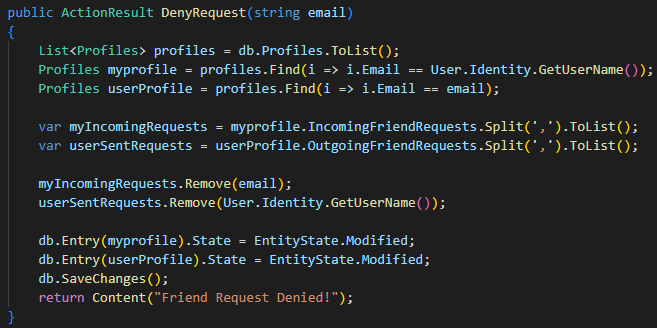
**Cancelling sent request.**



**Accepting Friend Request.**



**Denying Friend Request.**



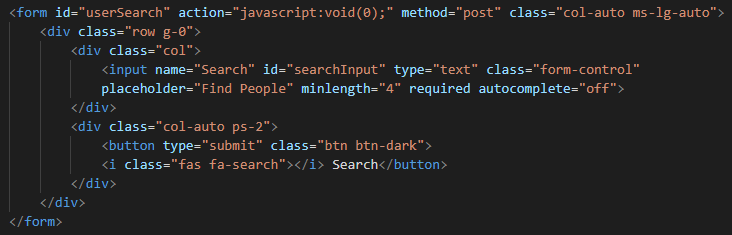
**Remove a user from Friend list.**



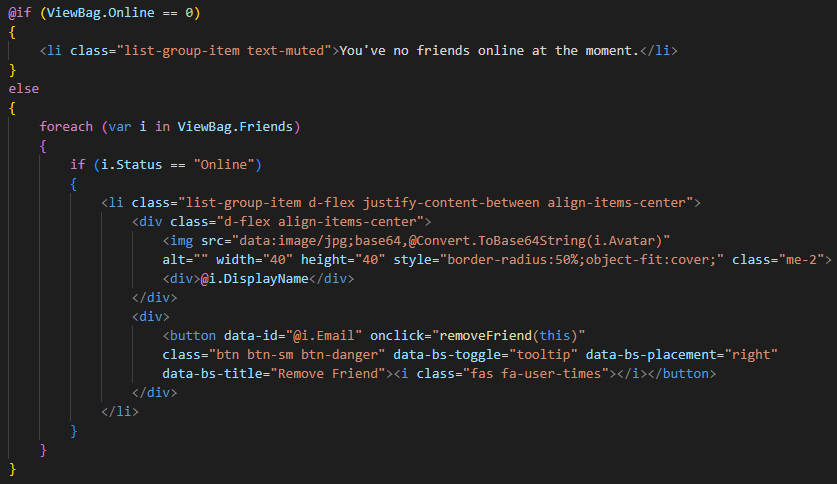
**Setting up Friends view page - Views/Friends/Index.cshtml.**

Similar to the Index page in Home, this also uses C#, HTML and JavaScript.

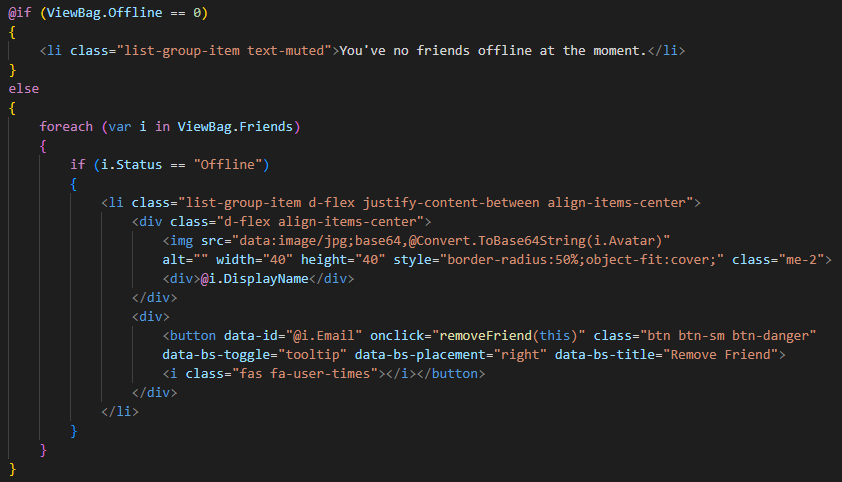
**User Search Form.**



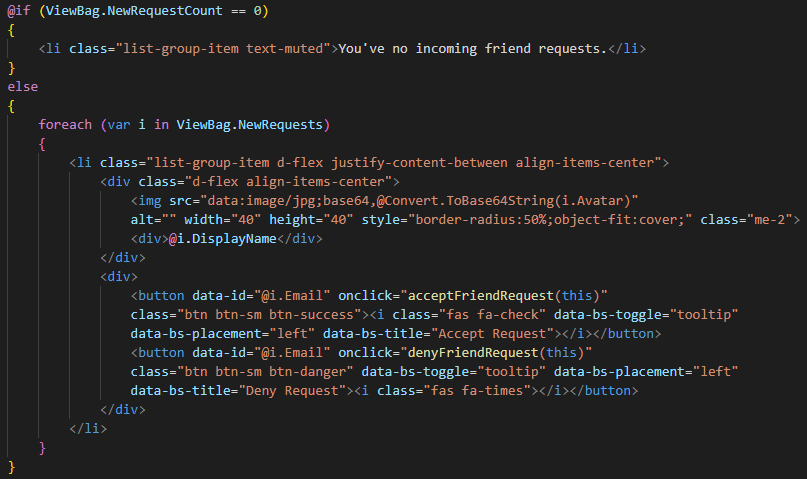
**Display list of Online Friends.**



**Display List of Offline Friends.**



**Display Incoming Friend Requests.**



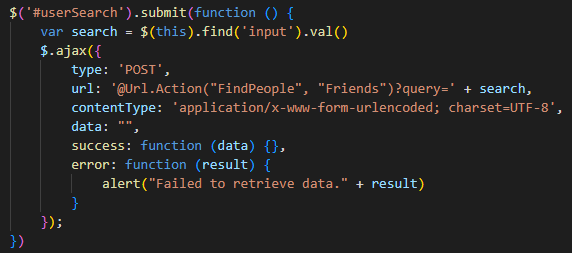
**Display list of sent requests.**



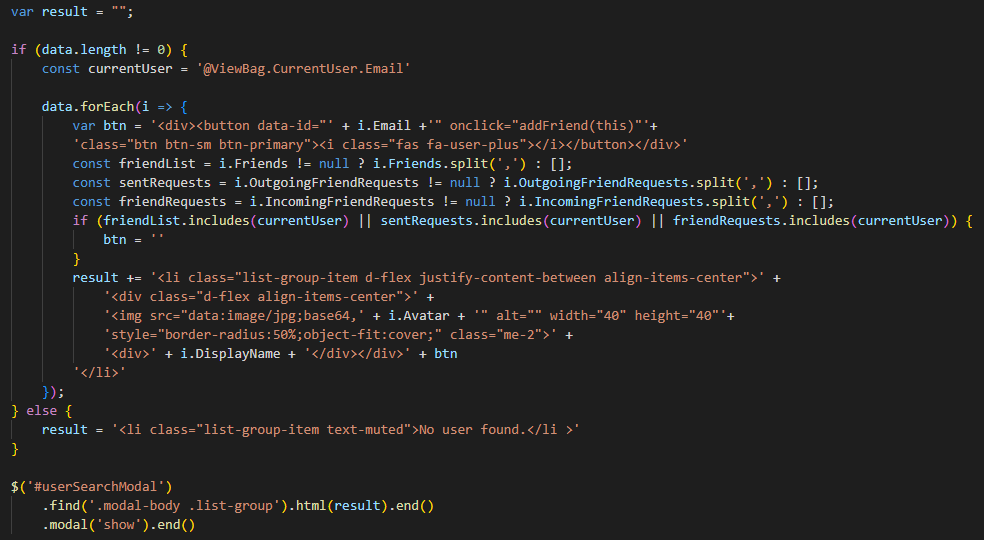
**JavaScript function for sending a new friend request.**



**Ajax function for searching a user in the database.**



On success:



**JavaScript function for cancelling a sent request.**



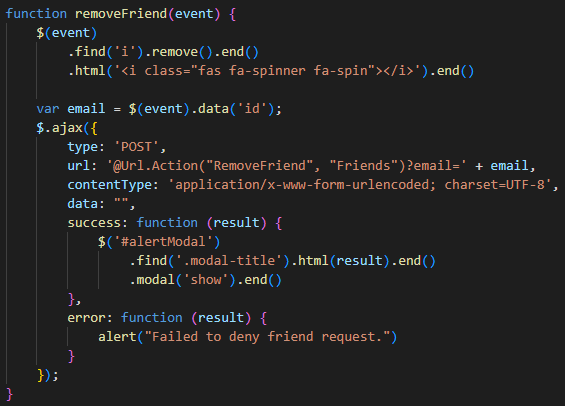
**JavaScript function for accepting a friend request.**



**JavaScript function for denying friend request.**



**JavaScript function for removing a user from friend list.**

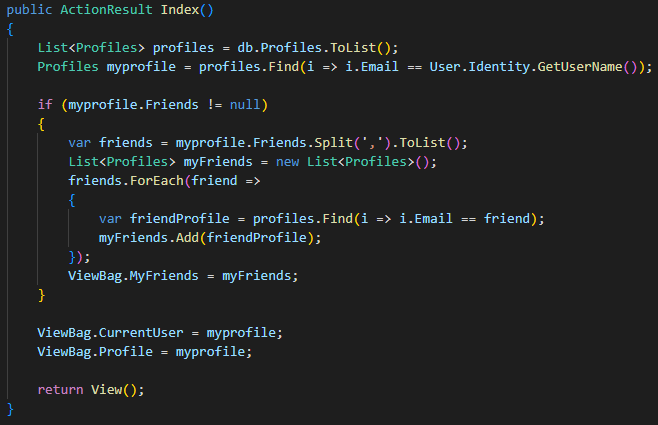


### Direct Messages

**Setting up DirectMessagesController.cs**.

Direct Messages is connected and based on friend system. User can only send direct messages to those who are added in their friend list.

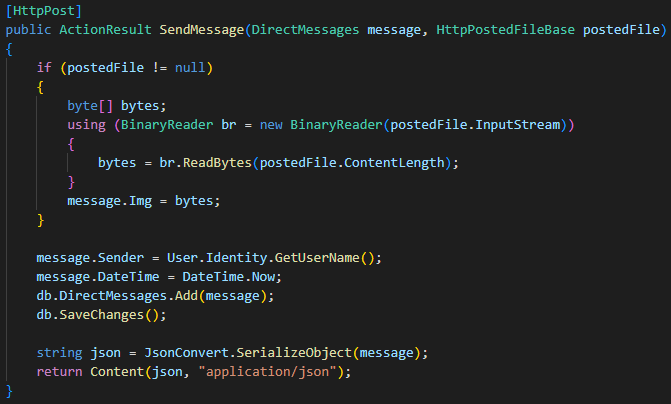
Function to retrieve all users from friend list.



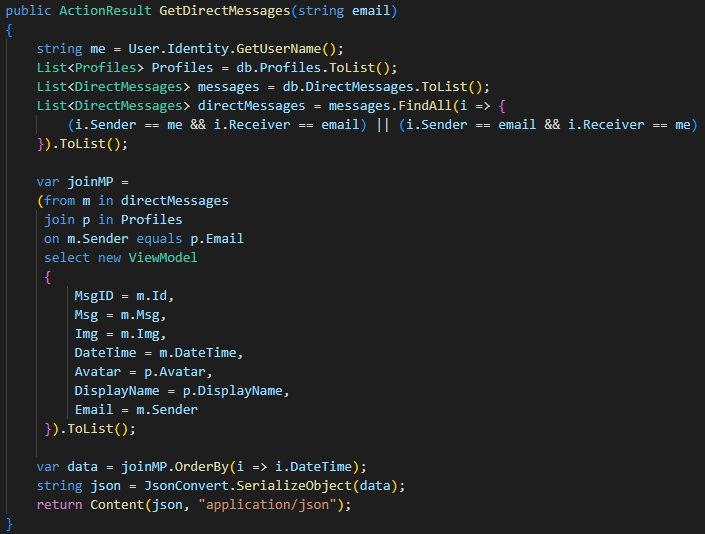
This function will show all users to whom the current user can send private messages.

Next up we have retrieving and sending messages just like Lobby except they are limited to user’s friend circle.

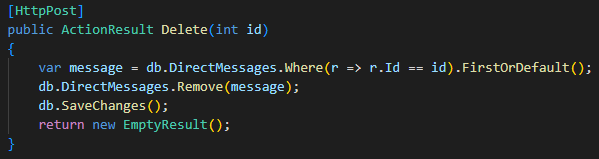
**Send Direct Message.**



**Get whole conversation between two users.**



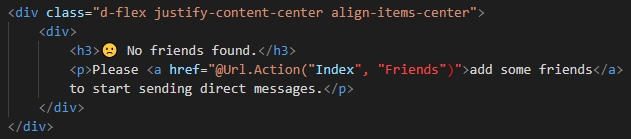
**Delete a private message.**



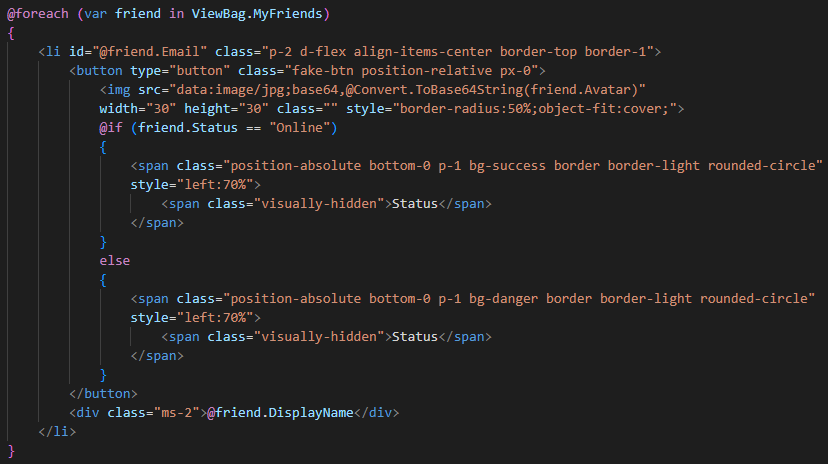
This function is same as the one defined in the Home Controller. It deletes an entry from the DirectMessages table.

**Setting up DirectMessages/Index.cshtml.**

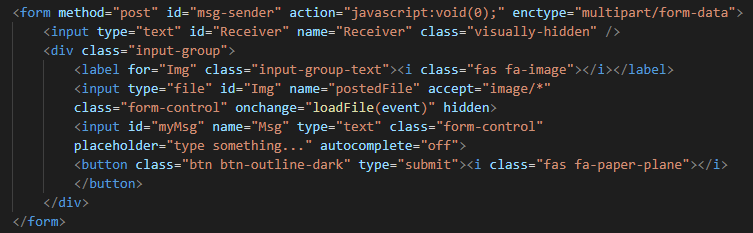
If the user’s friend list is empty, it will display the code below.



If the user has friends, this code will display the list of friends.



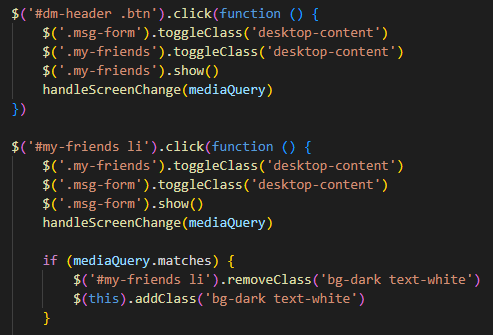
When the user clicks on a friend’s name, the page will display the send message form, similar to the one we have in Lobby.



The JavaScript part is same as for Lobby except it simply interacts with the DirectMessages Controller. Therefore, I’m going to skip adding JS codes here.

However, the design of DirectMessages page changes when on smaller screen devices. The screen size change movements are controlled from JavaScript.

Here’s the code for it. This function simply hides friends list and display chat screen. The rest is same as in the Lobby.



## Deployment

### Running or Debugging on a Single Machine

To debug the project, open the solution file from Visual Studio or directly from WebChat > WebChat.sln.



Use either of the buttons show in the image above to run the program without publishing.

A test account has been provided with the app to validate the connection.

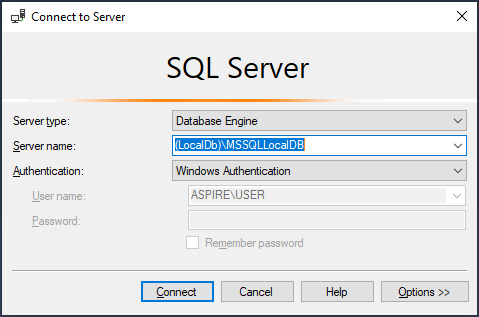
|  |  |  |
| --- | --- | --- |
| **Account Credentials** | | |
| **Email** | **:** | **amaan@mail.com** |
| **Password** | **:** | **Amaan@12345** |

Use this account to log in and test if you are successfully connected to SQL server.

If logged in successfully, the database should be created automatically inside your SQL Server LocalDb.

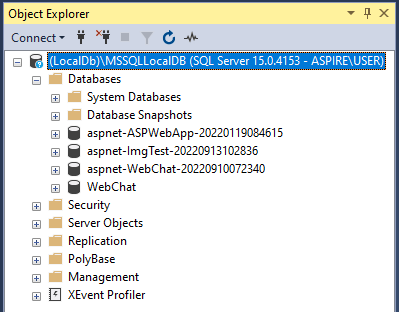
To access the database from SQL Server, open SSMS (SQL Server Management Studio) and connect to the server using the credentials given below.

|  |  |  |
| --- | --- | --- |
| **Server Name** | **:** | **(LocalDb)\MSSQLLocalDB** |
| **Authentication** | **:** | **Windows Authentication** |



Click on “Connect” to connect to the server.

You must see a database named “WebChat” under Databases inside the Object Explorer.



(SQL Server, 2022)

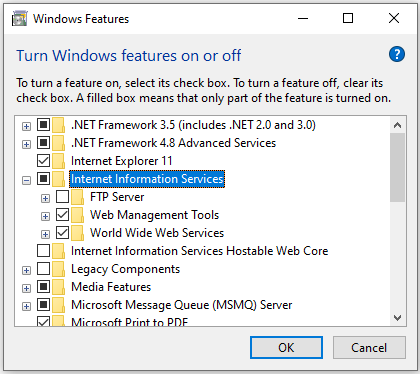
(Web Development with ASP.NET MVC and Core, 2022)

### Deploying on IIS (Local Network)

**# Step 1. Enable IIS on Windows.**

Open **Control Panel** > **Programs** > **Programs and Features** > **Turn Windows features on or off**.

Expand **Internet Information Services** and tick **Web Management Tools** and **World Wide Web Services** folder and all subfolders inside them.



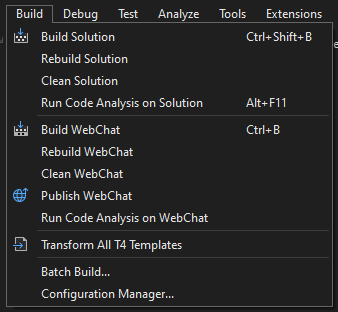
Select **OK**. A Dialog box indicating installation is taking place should appear.

Run the IIS Manager to verify the installation was successful.

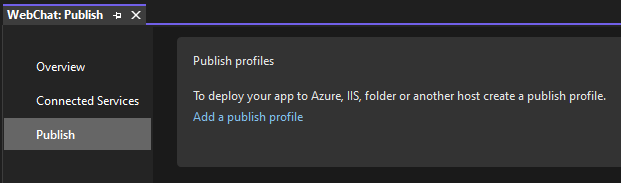
(Dykstra, 2022)

**# Step 2. Publish the web application to IIS folder using Visual Studio.**

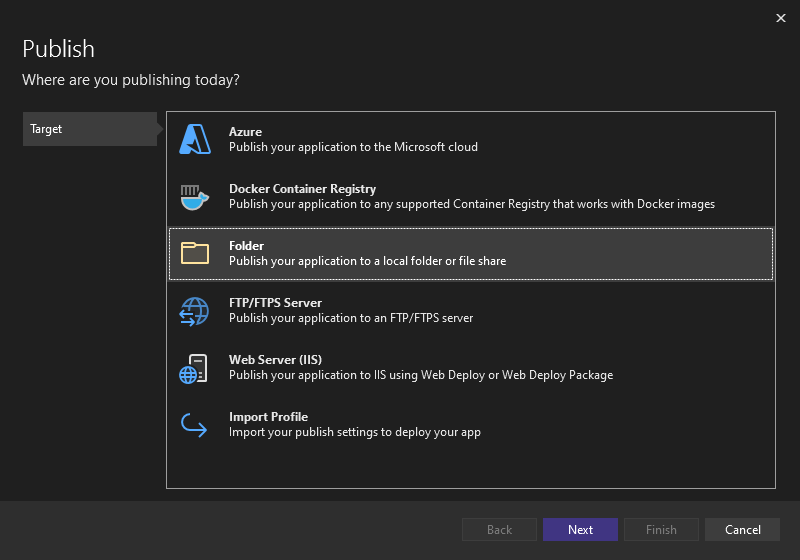
In Solution Explorer, right-click the project and select “Publish” or click on “Build” on the top menu and select “Publish WeChat.”



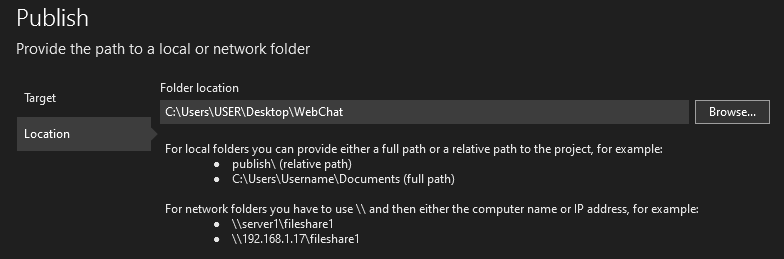
If you're publishing for the first time, you should see the Publish wizard open automatically. If not, click on “Add a publish profile” inside the Publish tab.



Then select the Folder option and click on Next.

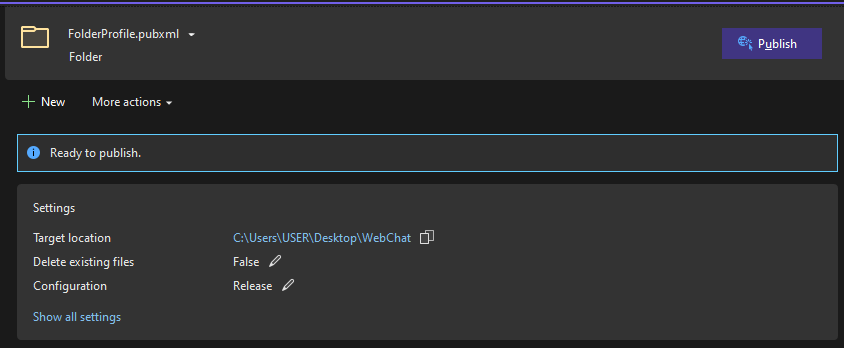


In the next screen, choose or create a new folder and click on Finish.



Due to permission issues, we cannot publish the app directly to the IIS folder.

Once done, click on the Publish button to publish the application on the given location.



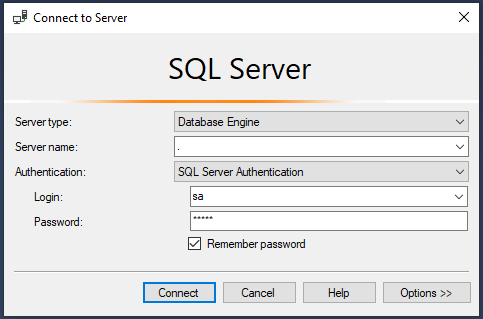
Now copy and paste the published folder inside IIS folder.

Go to **C:\inetpub\wwwroot** and paste the published folder here. Also copy and replace the **iisstart.htm** file given with this documentation in the same directory.

(Mikejo5000, 2022)

**# Step 3. Create a database in SQL Server.**

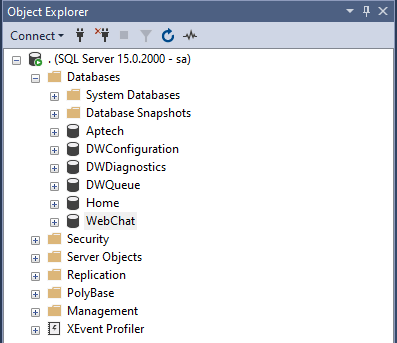
Open Microsoft SQL Server Management Studio (SSMS) and connect to “.” (period) server using SQL Server Authentication.



Use the username and password we created when installing SQL Server to login.

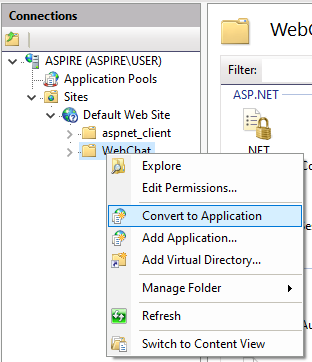
Open and execute the SQL query file named “Database-SQL-Query.sql” which is given with this documentation. You can also use the F5 button on your keyboard to run the query.

If the commands were completed successfully, you should see a new Database with the name “WebChat” created under databases folder in Object Explorer.



**# Step 4. Configuring IIS settings using IIS Manager.**

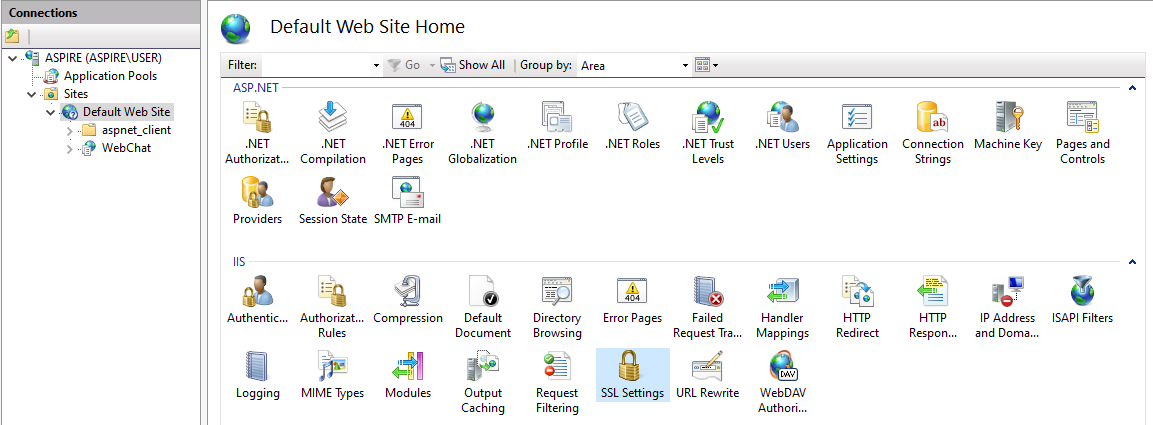
Open IIS Manager and expand your **Server** > **Sites** > **Default Web Site** > right-click on **WebChat** and select **Convert to Application**.

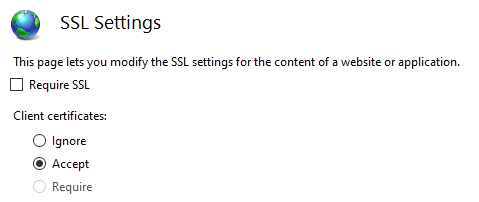


After converting to application. It should look like this.

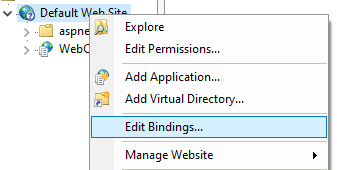


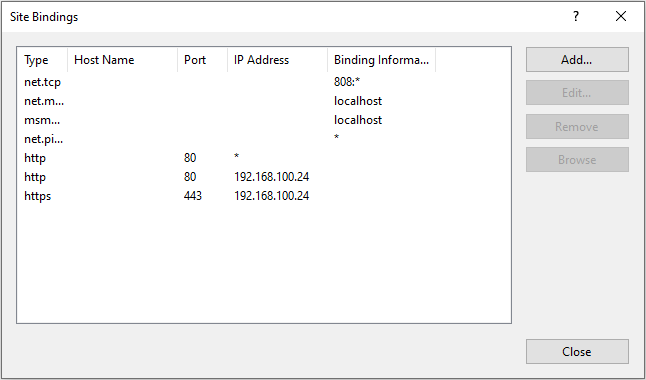
Now click on **Default Web Site** > double-click on **SSL Settings** > choose **Accept**.





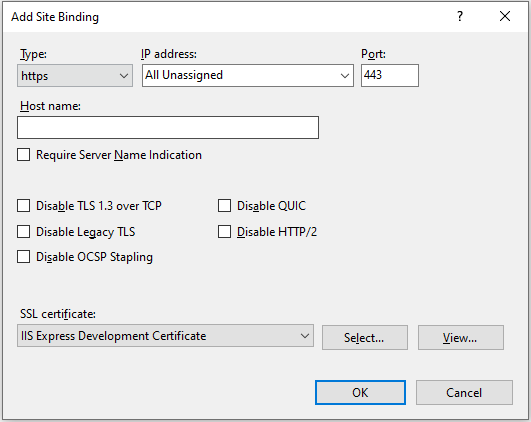
Next right-click on Default Web Site > Edit Bindings > Add.





New site binding settings are as following:

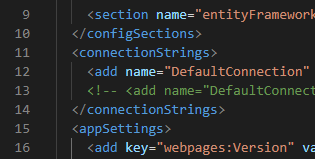
|  |  |
| --- | --- |
| **Type** | https |
| **SSL Certificate** | IIS Express Development Certificate |



(Dykstra, 2022)

**# Step 5. Configuring Web.config file.**

Go to **C:\inetpub\wwwroot\WebChat** and open **Web.config** with a code editor or notepad and look for **connectionStrings** at line 11.



Remove or comment the first connection and uncomment the second connection by removing **<!--** and **-->** from starting and ending of the line.

Your connection string should like this if you followed the steps correctly.



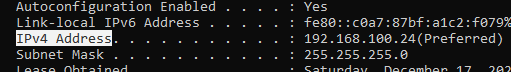
Replace “12345” with your password.



**# Step 6. Configuring windows hosts file for external logins.**

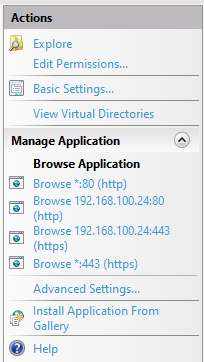
Retrieving your device IPv4 Address. To do this, open Command Prompt (CMD) and type **ipconfig /all** and hit enter.

Scroll down and look for **IPv4 Address** or simply search using CTRL+F.

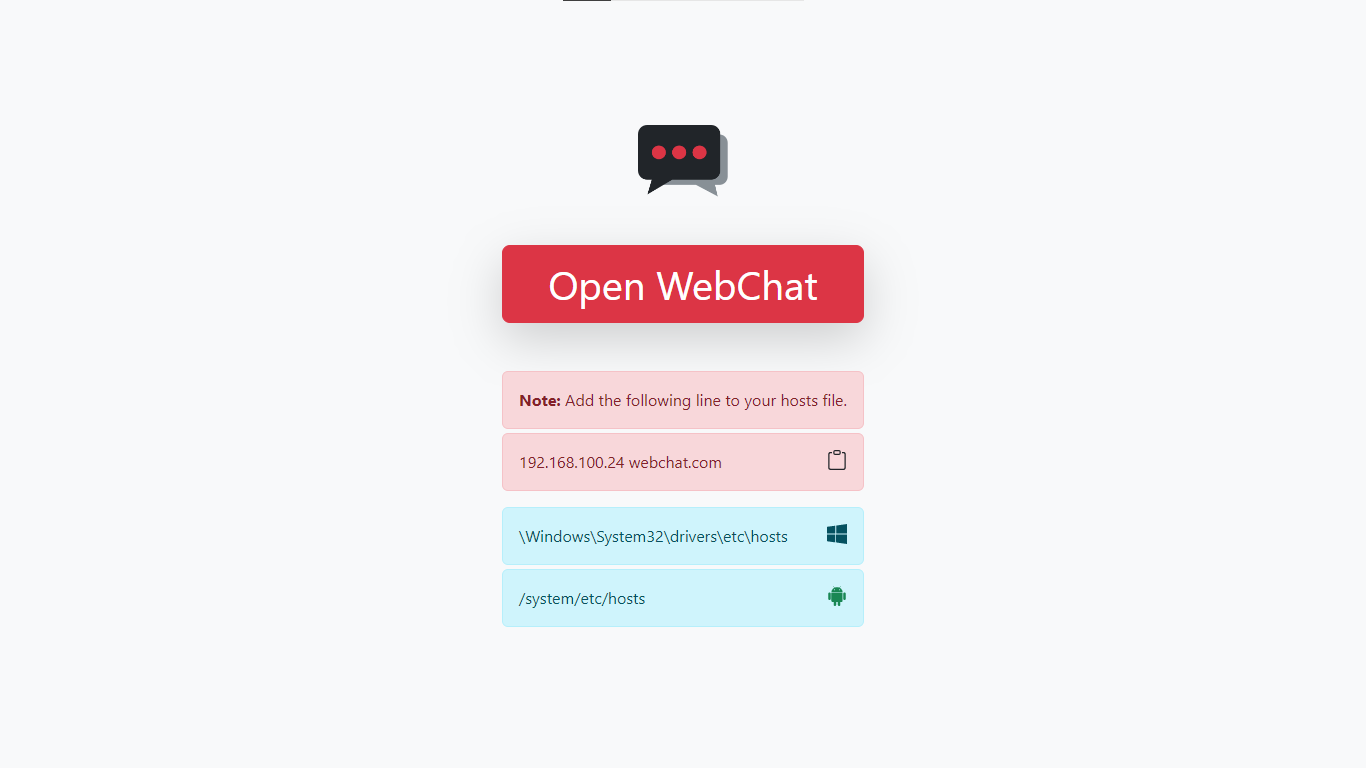


Note down the IP address given here.

Open a browser and go to **localhost** or use the IP address found above or simply browse the app from IIS Manager.



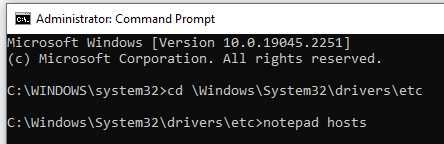
You should see this page.



Now open Command Prompt (CMD) with administrator rights and run the following two commands.

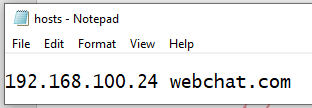
**cd \Windows\System32\drivers\etc**

**notepad hosts**



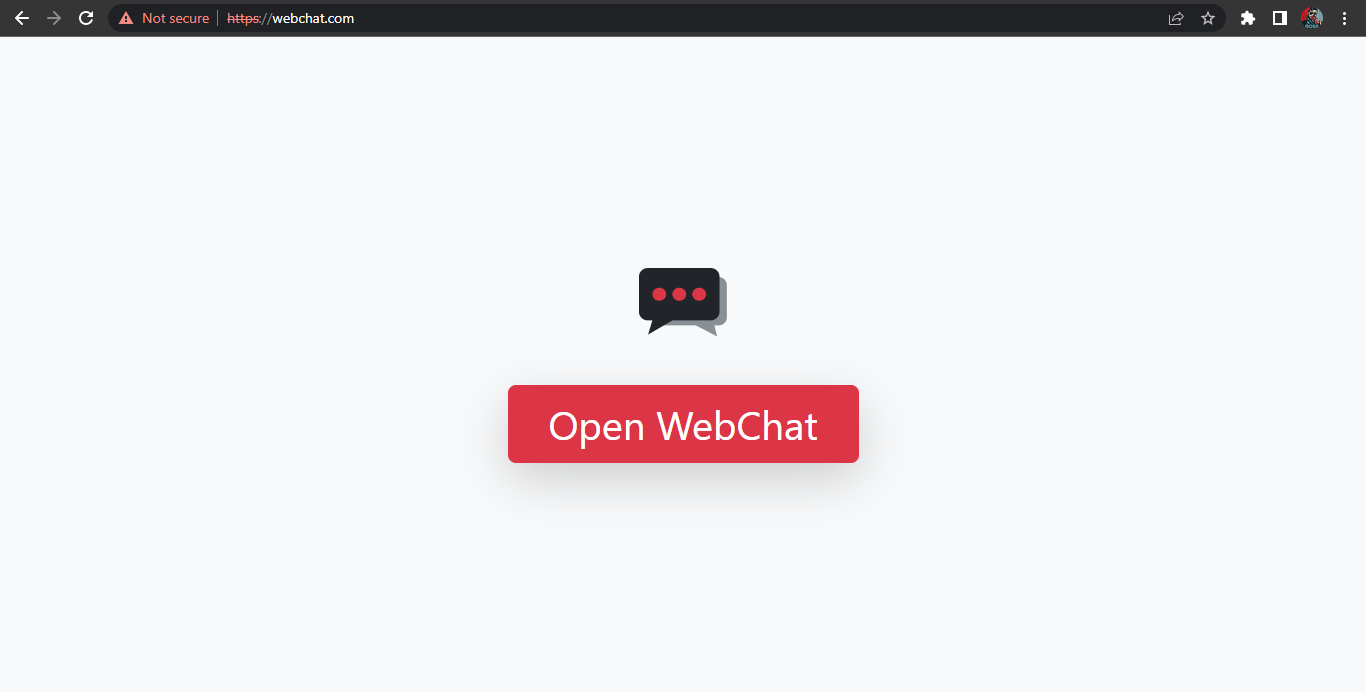
This should open the windows hosts file on a notepad.

Now add “YOUR\_IPv4\_ADDRESS webchat.com” at the very bottom in a new line without the double quotes.



Now you should be able to access the website by using “webchat.com”.

**Note:** You must perform step 6 on all devices you want to access the site using webchat.com. You won’t be able to use external logins like Google when accessing the site using IP address.



# Conclusion

The main goal of this project was to create secure and private communication system for small groups of people where they can feel relaxed and talk about their daily life without worrying about the outer world.

It was a wonderful and learning experience for me while working on this project. This project took me through the various stages of a business project development and gave me real insight into the world of web app development. The joy of work and the thrill involved while tackling numerous problems and challenges gave me a feel of developer’s industry.

I enjoyed each and every bit of work I had put into this project.

# References

*ASP.NET*. (n.d.). Retrieved from Microsoft: http://asp.net/

Beat, S. (n.d.). *Website Redesign Proposal Template*. Retrieved from Visme: https://www.visme.co/templates/proposals/website-redesign-proposal-1425285156/

*Discord*. (n.d.). Retrieved from Discord: https://discord.com/

Dykstra, T. (2022, June 30). *ASP.NET Web Deployment using Visual Studio: Deploying to Test.* Retrieved from Microsoft: https://learn.microsoft.com/en-us/aspnet/web-forms/overview/deployment/visual-studio-web-deployment/deploying-to-iis

(2023). *Final Report Structure In Computing Project.* NCC Education Limited.

Mikejo5000. (2022, April 29). *Publish an ASP.NET web app.* Retrieved from Microsoft: https://learn.microsoft.com/en-us/visualstudio/deployment/quickstart-deploy-aspnet-web-app

Otto, M. (2022, November 22). *Bootstrap Documentation*. Retrieved from Bootstrap: https://getbootstrap.com/docs/5.2/

Polesny, O. (2021, January 18). *Debounce – How to Delay a Function in JavaScript (JS ES6 Example)*. Retrieved from freeCodeCamp: https://www.freecodecamp.org/news/javascript-debounce-example/

*SQL Server*. (2022). Retrieved from Microsoft: https://www.microsoft.com/sql-server/

*Web Development with ASP.NET MVC and Core.* (2022). Doha: Aptech Limited.

Wick, J. (n.d.). *Web Development Project Proposal Template*. Retrieved from Visme: https://www.visme.co/templates/proposals/web-development-project-proposal-1425280046/