İstanbul Okan Üniversitesi Bilgisayar Mühendisliği Bölümü

CENG 216

COMPUTER NETWORKS

SPRING 2024

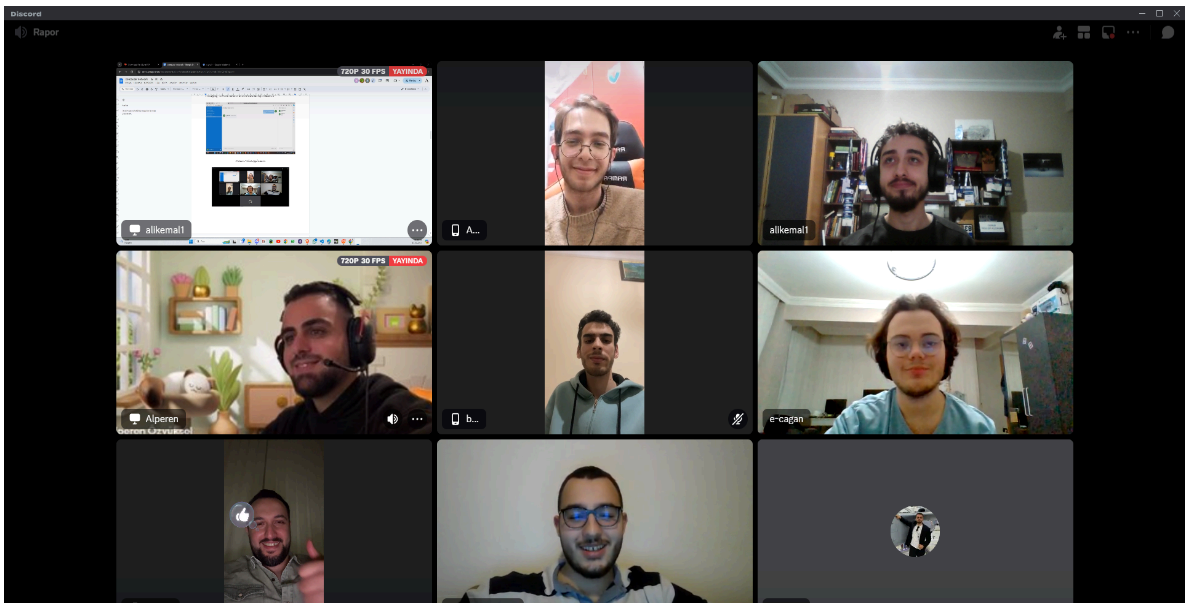
SEMESTER PROJECT

Real-time Instant Messaging Infrastructure

Contents

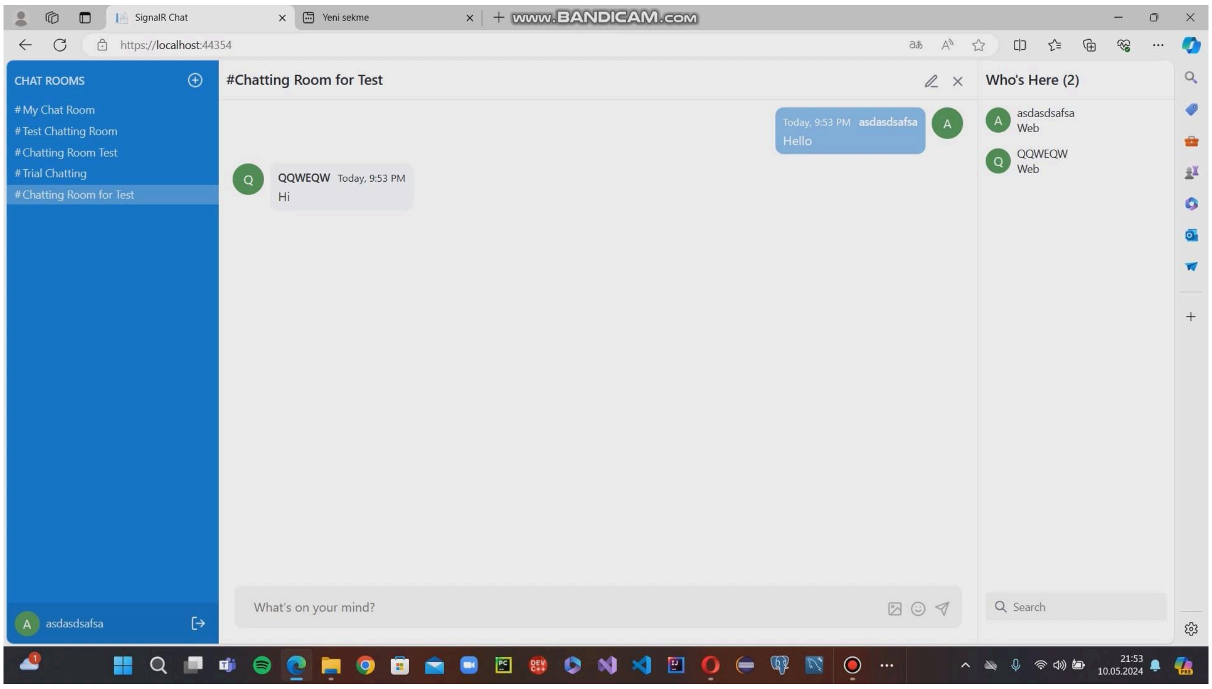
1-Project Team

Member ID	Student ID	First Name	Last Name
1	210212032	Alperen	Özyüksel
2	210212042	Bercan	Aydın
3	220212054	Emin Çağan	Apaydın
4	210212031	Ali Kemal	Bayram
5	210212051	Atakan	Onay
6	210212016	Emir Ali	Öner



2-Project Requests

The aim of the project is to create a real-time messaging application. This app will allow two or more people to message each other. In addition to text-based messaging, users will be able to share files among themselves.



Today, 9:53 PM asdasdsafsa
Hello

QQWEQW Today, 9:53 PM
Hi

Picture 1 Chat Application

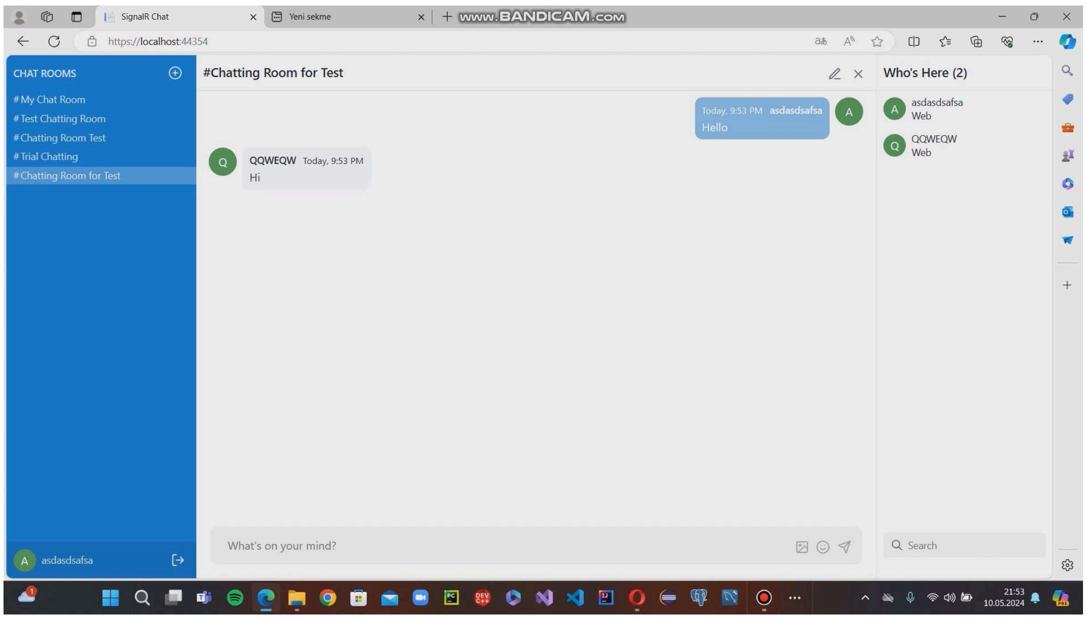
3- Objective

This project has two main objectives. Firstly, to acquire knowledge in the areas of network communication (establishing a connection over a network), sockets, I/O (text-based, real-time communication), and threads (two or more people communicating simultaneously). Particularly, to improve our skills in the SignalR and Knockout.js libraries we use. To familiarize ourselves with the web and .NET 7 world.

Secondly, to design a functional and aesthetic interface for the chat system.

4-Conversations

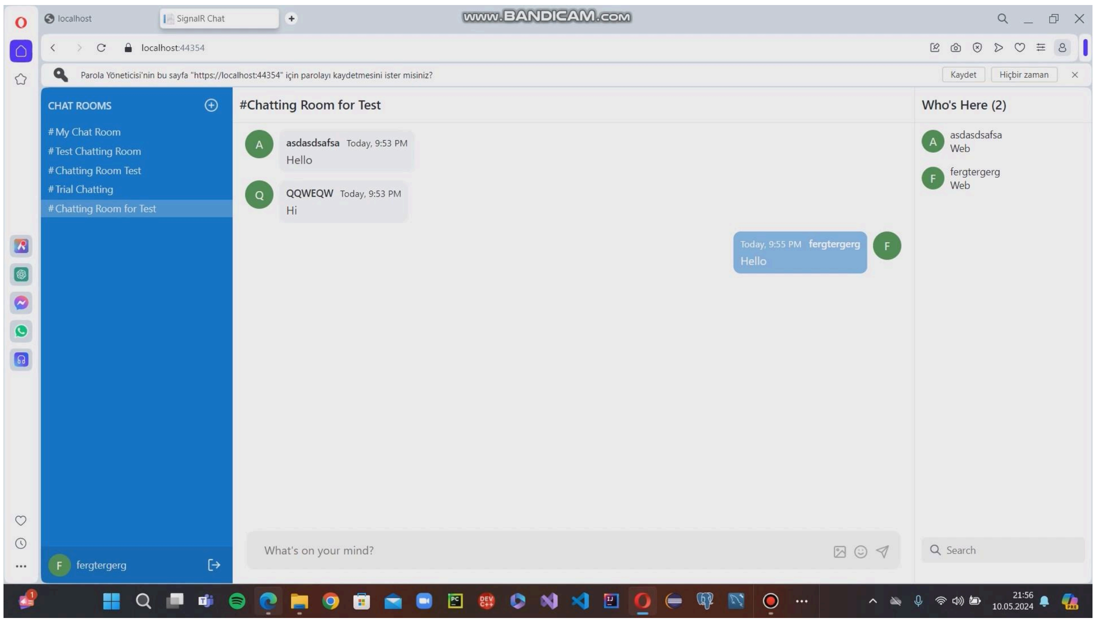
Users can join various groups or engage in one-on-one conversations. The application also allows for anonymous messaging and enables users to see who is online



Today, 9:53 PM asdasdsafsa
Hello

QQWEQW Today, 9:53 PM
Hi

Picture 2 Chat with one person



Parola Yöneticisi'nin bu sayfa "https://localhost:44354" için parolayı kaydetmesini ister misiniz?

CHAT ROOMS

My Chat Room
Test Chatting Room
Chatting Room Test
Trial Chatting
Chatting Room for Test

#Chatting Room for Test

asdasdsafsa Today, 9:53 PM
Hello

QQWEQW Today, 9:53 PM
Hi

Today, 9:55 PM fergtergerg
Hello

What's on your mind?

Who's Here (2)

asdasdsafsa
Web

fergtergerg
Web

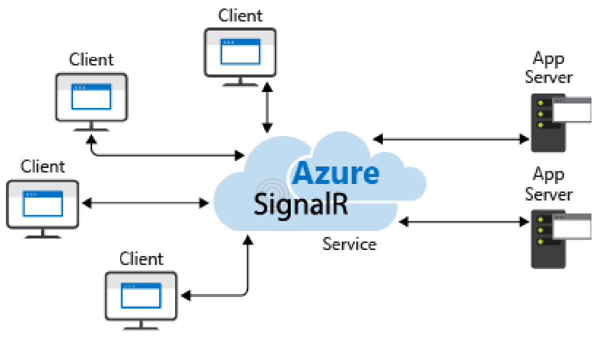
fergtergerg

Picture 3 Group chat

5-Client/Server Interaction

In the research conducted to enable simultaneous communication between the client and server, the SignalR library was preferred.

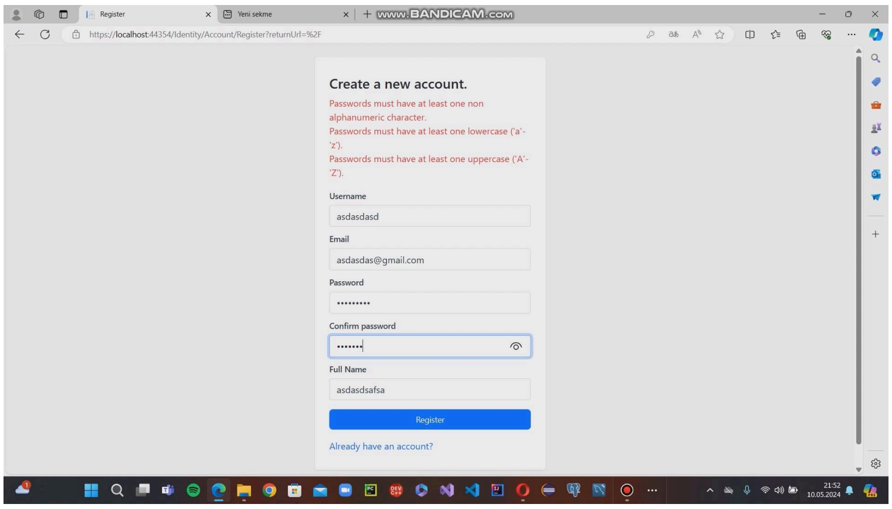
The SignalR service forwards the input received from the client to the relevant server and broadcasts the response to all clients, thereby enabling simultaneous communication.



Picture 4 SignalR Working Principle

6-Authentication

To use the chat application, an account must be created. The user chooses a username and password. The password must adhere to various security rules.



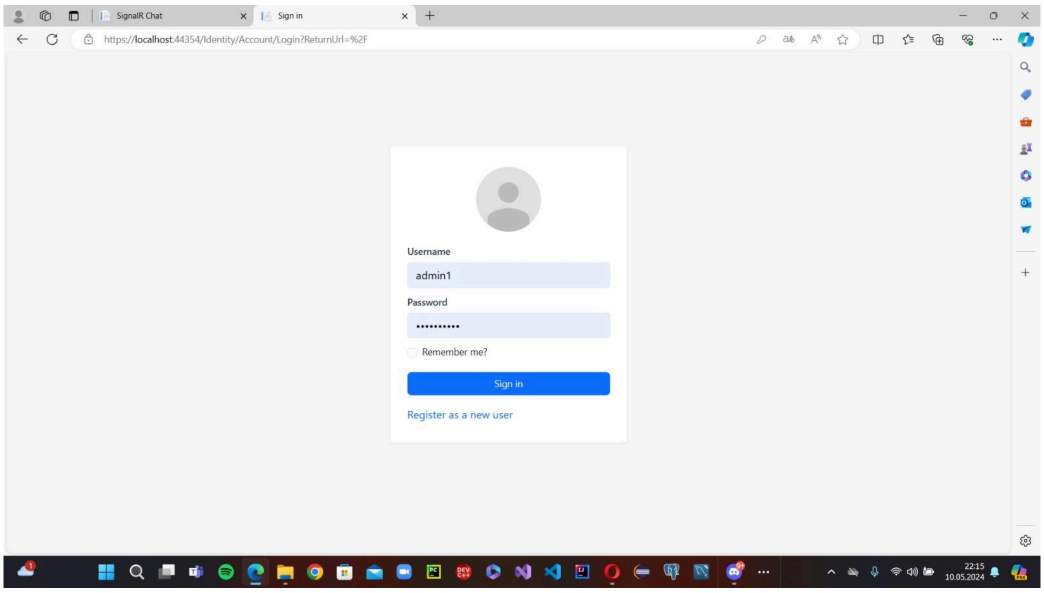
https://localhost:44354/Identity/Account/Register?returnUrl=%2F

Create a new account.

Passwords must have at least one non alphanumeric character.
Passwords must have at least one lowercase ('a'-'z').
Passwords must have at least one uppercase ('A'-'Z').

Username: asdasdasd
Email: asdasdas@gmail.com
Password: ••••••••••
Confirm password: •••••••
Full Name: asdasdsafsa

Picture 5 Creating Account



https://localhost:44354/Identity/Account/Login?ReturnUrl=%2F

Username: admin1
Password: ••••••••••
Remember me?

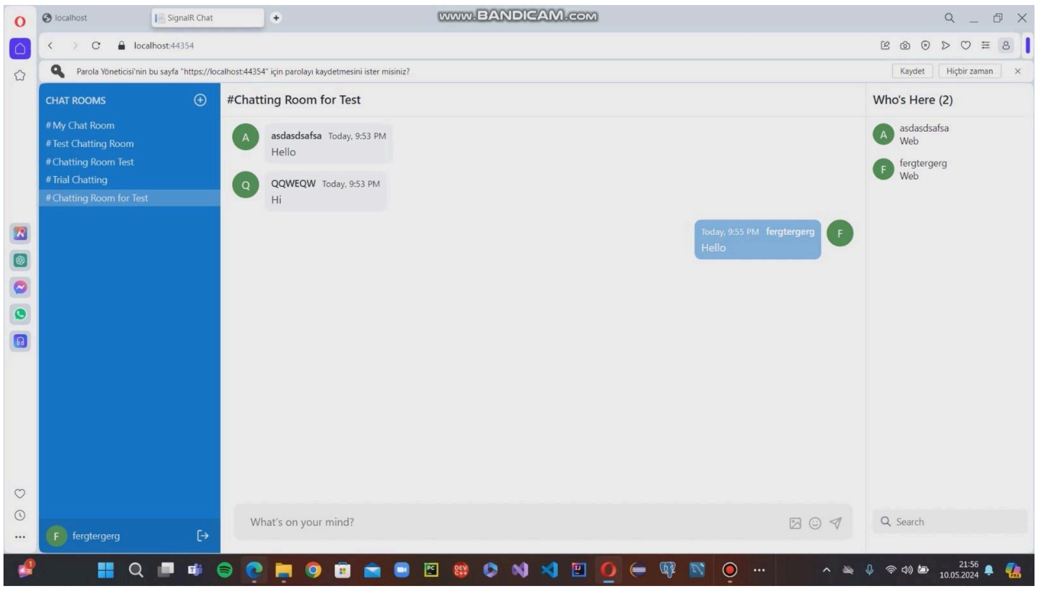
Picture 6 Password-based user login

7-Tasks of the Project

Real-time visibility of active users was provided. Chatrooms were created, allowing users to join and leave conversations. Users were allowed to participate in multiple conversations simultaneously. The ability to view the entire message history of a conversation was ensured as long as the user is engaged in the conversation. An option to create an account was established, depending on identity verification.

8-Usability design

The design is crafted to be user-friendly, while also staying true to the interface perception created by commonly used chat applications like WhatsApp and Telegram, which people are accustomed to in today's world.



Parola Yöneticisi'nin bu sayfa "https://localhost:44354" için parolayı kaydetmesini ister misiniz?

asdasdsafsa Today, 9:53 PM
Hello

QQWEQW Today, 9:53 PM
Hi

Today, 9:55 PM fergtergerg
Hello

Picture 7 Chat interface

9-Test strategy

Test strategy is extremely important because it determines what we need to do in testing. Therefore, the application was tested under various conditions through the web. The application was tested under various conditions on the web, and when the user traffic increased, it was observed and tested whether the system was functioning properly. Because the test strategy will take us further from where we are. Therefore, we were very careful when developing the test strategy. The test strategy also ensured that our communication program became much better.

10-Implementation

The application uses .NET 7, SignalR, and KnockoutJS for the chat application. The application operates via a web interface.

11-Outcome

As a result, by using Python websockets, a real-time messaging application was coded, and the functioning of websockets was learned.

11.1 What was easy?

Developing a simple chat application using WebSockets was comfortable due to the generally straightforward usage of WebSocket libraries. The usage of libraries that support WebSocket functionality in Python is quite clear and designed to adhere to a specific protocol. Additionally, the efficiency and speed of WebSockets in enabling real-time communication also facilitated the process.

11.2 What was difficult?

The challenging part of the project was managing WebSocket connections and enabling users to communicate in real-time. Especially ensuring multiple users communicate in the same chat room and handling the correct transmission and processing of messages can pose some technical challenges. Additionally, managing the user interface and dynamic updates on the client-side can also be demanding. However, overcoming all these challenges successfully, the project was completed.

11.3 What was unexpected?

There were no unexpected challenges encountered during the project process; however, it was important not to forget that WebSockets could sometimes lead to connection issues or timeouts. Handling such situations and providing user-friendly error messages could be crucial.

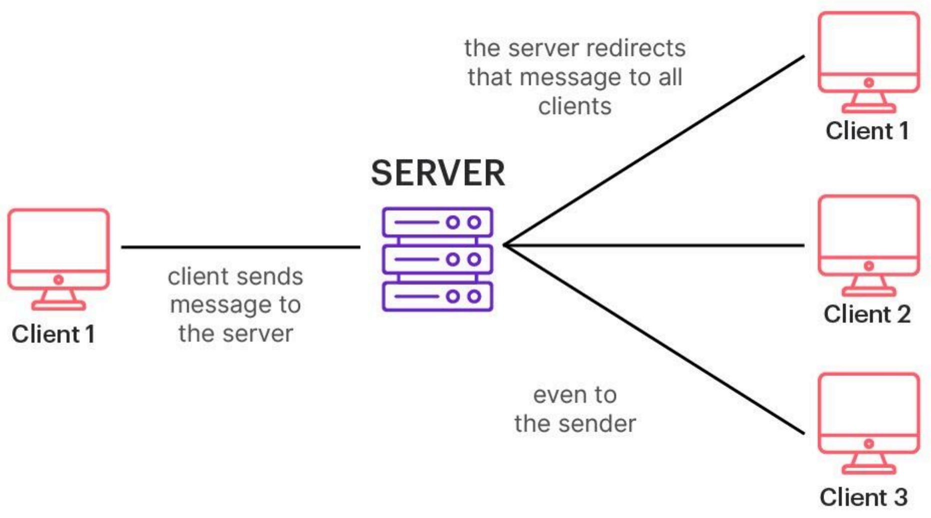
11.4 If it were to be done again, what should be paid attention to?

When designing the chat system, there are several things that can be done differently to further enhance the user experience. For example, creating a more advanced user interface to allow users to create or join chat rooms. Additionally, a more complex message management system could be added to help users better organize their messages.

Additionally, considering security measures and user authentication, adding advanced authorization and access controls will be important. Taking such measures is necessary to protect user data and ensure a secure communication environment.

12- To Learn Real-Time Chatting Theory

During the project process, the theory of real-time messaging was successfully learned. Real-time messaging is an important communication method that enables users to communicate in real-time from different geographical locations. During this process, the basic principles and operation of real-time messaging were understood. How real-time communication is achieved, synchronized interaction, and the possibilities of multi-party communication were learned. Technologically, understanding the infrastructure of real-time messaging was central to the project. By learning communication protocols and server-client architectures, it was understood how real-time messaging applications work.

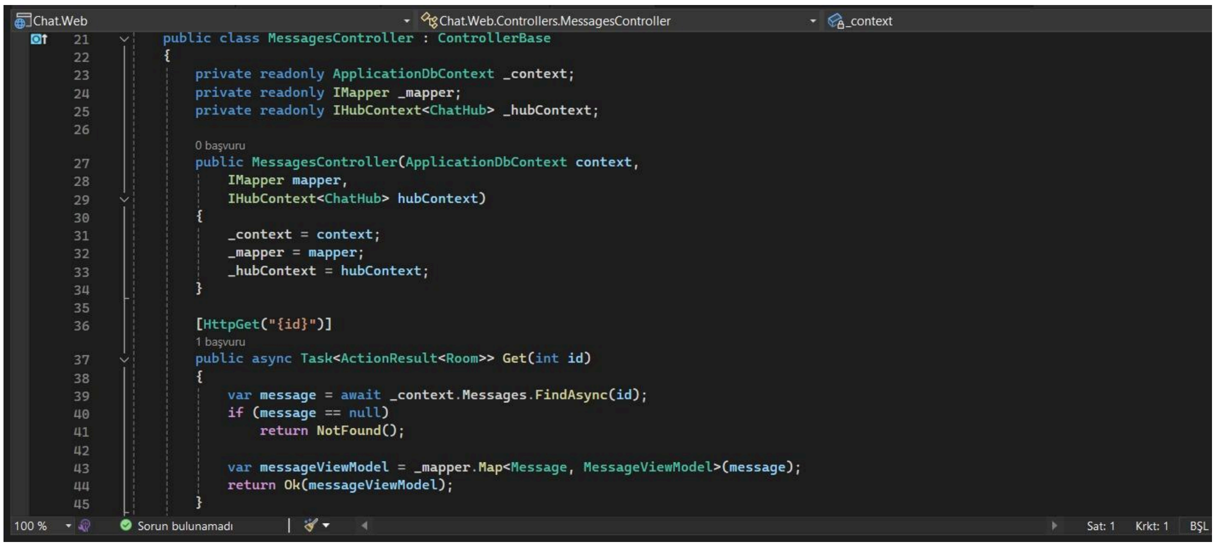

the server redirects that message to all clients
Client 1
SERVER
Client 1
client sends message to the server
Client 2
even to the sender
Client 3

Picture 8 Real-Time Communication

13-Deciding on the Ready Application to Simulate the Project with

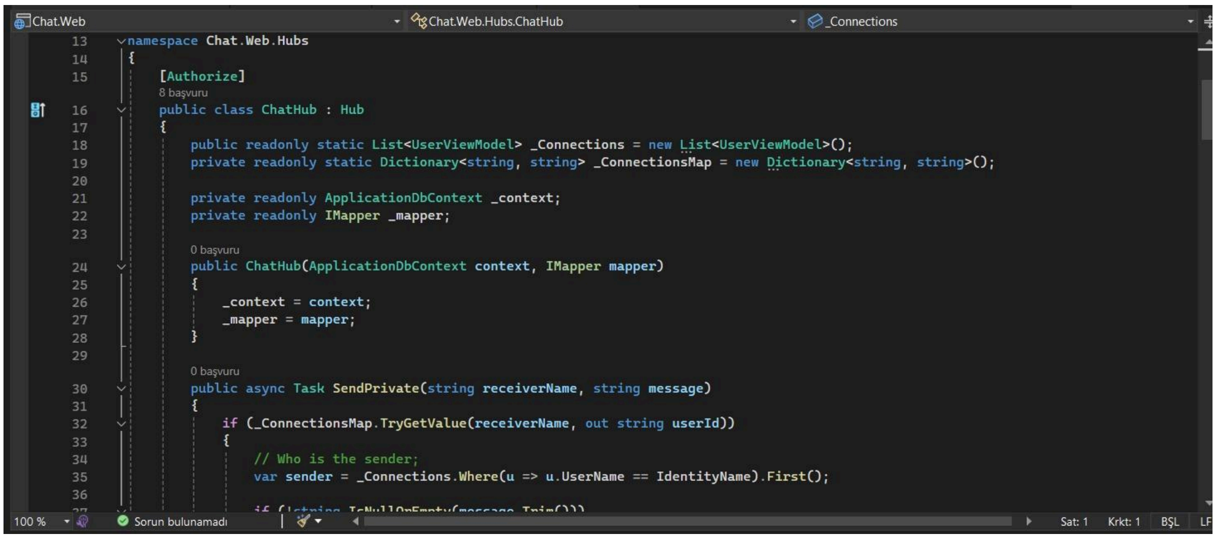
The most suitable decision for the project was made by examining WhatsApp Web and similar applications. This process was an important step in the development of the project and helped to better understand the features and working principles of the application.

14-Pictures From Coding



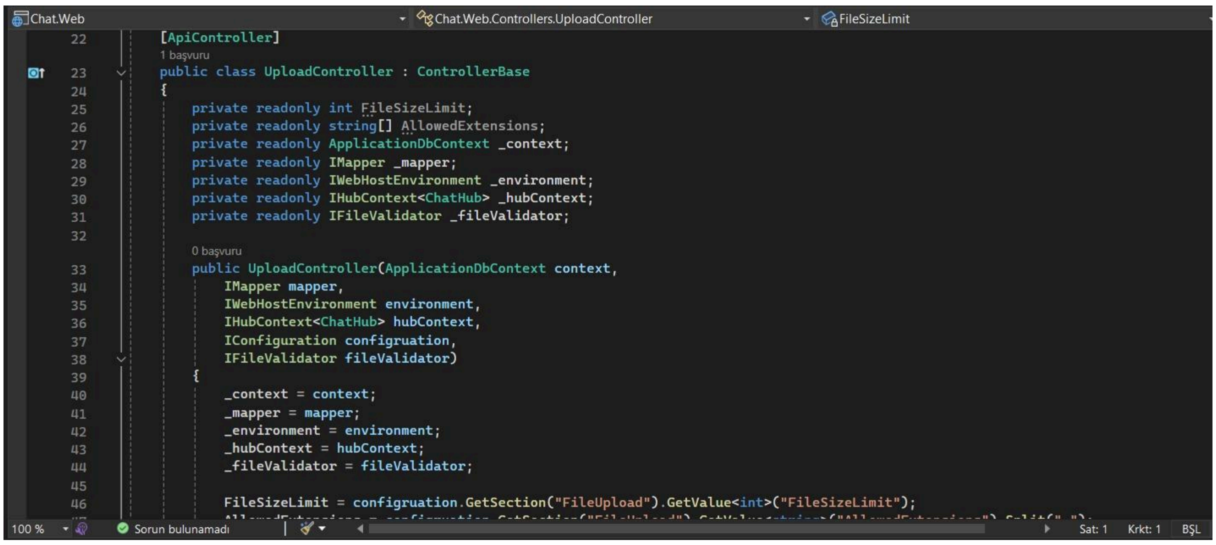
```
public class MessagesController : ControllerBase
{
    private readonly ApplicationDbContext _context;
    private readonly IMapper _mapper;
    private readonly IHubContext<ChatHub> _hubContext;

    0 başvuru
    public MessagesController(ApplicationDbContext context,
        IMapper mapper,
        IHubContext<ChatHub> hubContext)
    {
        _context = context;
        _mapper = mapper;
        _hubContext = hubContext;
    }

    [HttpGet("{id}")]
    1 başvuru
    public async Task<ActionResult<Room>> Get(int id)
    {
        var message = await _context.Messages.FindAsync(id);
        if (message == null)
            return NotFound();

        var messageViewModel = _mapper.Map<Message, MessageViewModel>(message);
        return Ok(messageViewModel);
    }
```

Picture 9 Massage Controller



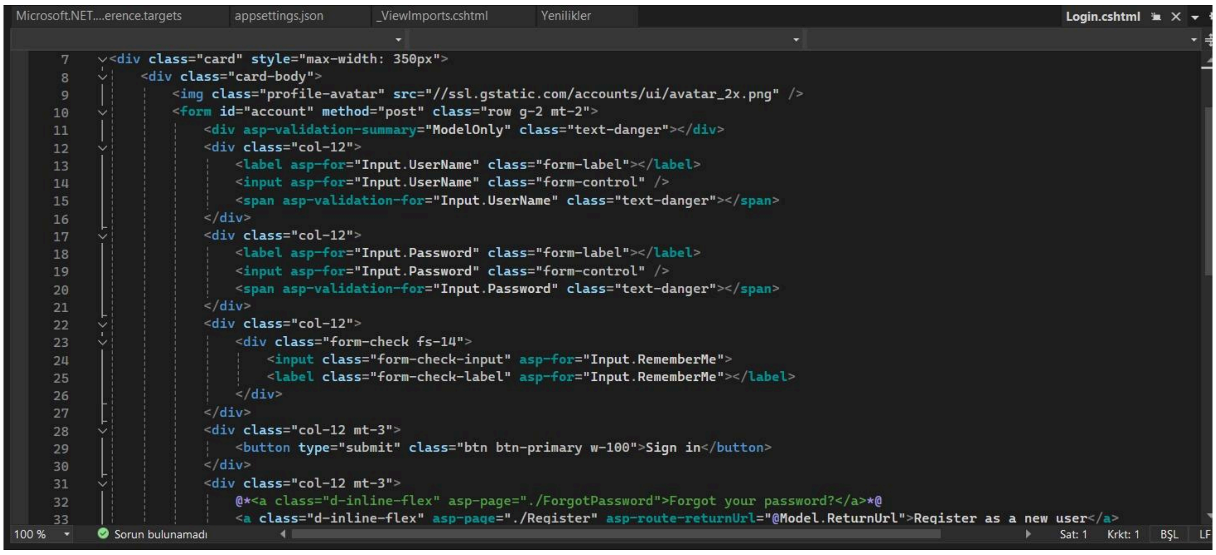
```
namespace Chat.Web.Hubs
{
    [Authorize]
    8 başvuru
    public class ChatHub : Hub
    {
        public readonly static List<UserViewModel> _Connections = new List<UserViewModel>();
        private readonly static Dictionary<string, string> _ConnectionsMap = new Dictionary<string, string>();

        private readonly ApplicationDbContext _context;
        private readonly IMapper _mapper;

        0 başvuru
        public ChatHub(ApplicationDbContext context, IMapper mapper)
        {
            _context = context;
            _mapper = mapper;
        }

        0 başvuru
        public async Task SendPrivate(string receiverName, string message)
        {
            if (_ConnectionsMap.TryGetValue(receiverName, out string userId))
            {
                // Who is the sender;
                var sender = _Connections.Where(u => u.UserName == IdentityName).First();

```

Picture 10 Chatting Center



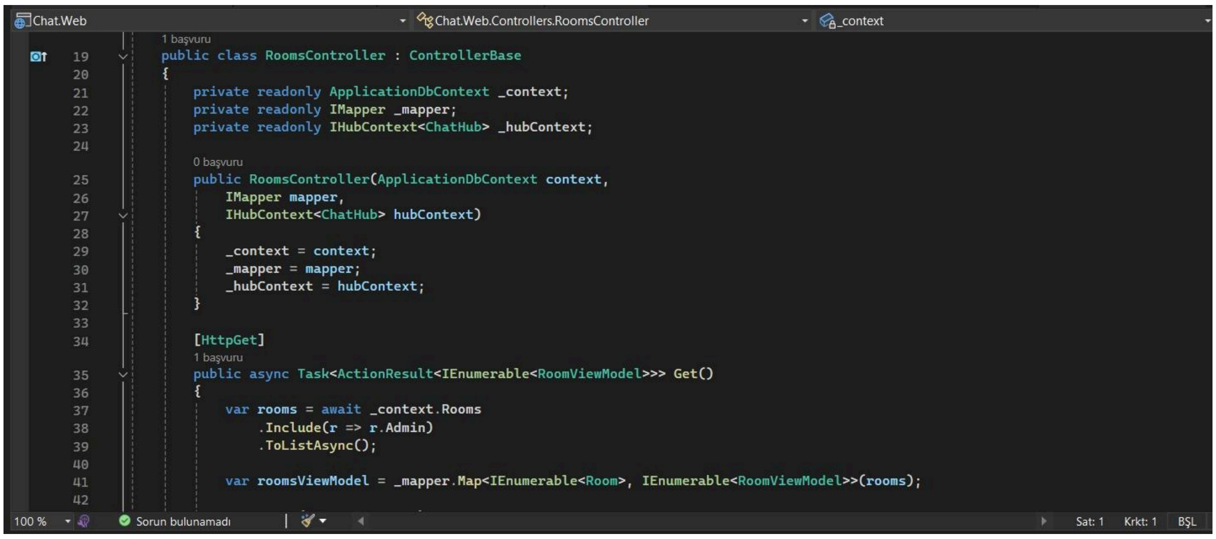
```
    [ApiController]
    1 başvuru
    public class UploadController : ControllerBase
    {
        private readonly int FileSizeLimit;
        private readonly string[] AllowedExtensions;
        private readonly ApplicationDbContext _context;
        private readonly IMapper _mapper;
        private readonly IWebHostEnvironment _environment;
        private readonly IHubContext<ChatHub> _hubContext;
        private readonly IFileValidator _fileValidator;

        0 başvuru
        public UploadController(ApplicationDbContext context,
            IMapper mapper,
            IWebHostEnvironment environment,
            IHubContext<ChatHub> hubContext,
            IConfiguration configruation,
            IFileValidator fileValidator)
        {
            _context = context;
            _mapper = mapper;
            _environment = environment;
            _hubContext = hubContext;
            _fileValidator = fileValidator;

            FileSizeLimit = configruation.GetSection("FileUpload").GetValue<int>("FileSizeLimit");
```

Picture 11 Loading Controller



```
<div class="card" style="max-width: 350px">
    <div class="card-body">
        <img class="profile-avatar" src="//ssl.gstatic.com/accounts/ui/avatar_2x.png" />
        <form id="account" method="post" class="row g-2 mt-2">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="col-12">
                <label asp-for="Input.UserName" class="form-label"></label>
                <input asp-for="Input.UserName" class="form-control" />
                <span asp-validation-for="Input.UserName" class="text-danger"></span>
            </div>
            <div class="col-12">
                <label asp-for="Input.Password" class="form-label"></label>
                <input asp-for="Input.Password" class="form-control" />
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="col-12">
                <div class="form-check fs-14">
                    <input class="form-check-input" asp-for="Input.RememberMe">
                    <label class="form-check-label" asp-for="Input.RememberMe"></label>
                </div>
            </div>
            <div class="col-12 mt-3">
                <button type="submit" class="btn btn-primary w-100">Sign in</button>
            </div>
            <div class="col-12 mt-3">
                @*<a class="d-inline-flex" asp-page="./ForgotPassword">Forgot your password?</a>*@
                <a class="d-inline-flex" asp-page="./Register" asp-route-returnUrl="@Model.ReturnUrl">Register as a new user</a>
```

Picture 12 Login



```
1 başvuru
public class RoomsController : ControllerBase
{
    private readonly ApplicationDbContext _context;
    private readonly IMapper _mapper;
    private readonly IHubContext<ChatHub> _hubContext;

    0 başvuru
    public RoomsController(ApplicationDbContext context,
        IMapper mapper,
        IHubContext<ChatHub> hubContext)
    {
        _context = context;
        _mapper = mapper;
        _hubContext = hubContext;
    }

    [HttpGet]
    1 başvuru
    public async Task<ActionResult<IEnumerable<RoomViewModel>>> Get()
    {
        var rooms = await _context.Rooms
            .Include(r => r.Admin)
            .ToListAsync();

        var roomsViewModel = _mapper.Map<IEnumerable<Room>, IEnumerable<RoomViewModel>>(rooms);

```

Picture 13 Chatting room controller

15-Source

https://learn.microsoft.com/tr-tr/aspnet/signalr/overview/getting-started/tutorial-getting-started-with-signalr

https://www.researchgate.net/publication/370516068_Real-Time_Chat_Application

https://www.researchgate.net/profile/Sura-Khalaf/publication/377077969_New_Approach_for_Security_Chatting_in_Real_Time/links/6594544b0bb2c7472b2c451b/New-Approach-for-Security-Chatting-in-Real-Time.pdf

https://portal.bazeuniversity.edu.ng/student/assets/thesis/202102151206581490636642.pdf