# SignalR Chat Application with WebRTC Integration

## **Overview**

This documentation outlines the SignalR-based chat application, which supports real-time messaging and video calls between users. The application integrates ASP.NET Core with SignalR for real-time communication and uses **AuthService**, **ChatService**, and **UserService** for authentication, message handling, and user profile management.

## **Table of Contents**

1. [Prerequisites](#prerequisites)
2. [Application Structure](#application-structure)
   1. [Client-Side: JavaScript and SignalR](#client-side-javascript-and-signalr)
   2. [Server-Side: SignalR Setup](#server-side-signalr-setup)
   3. [Login Page Configuration](#login-page-configuration)
   4. [WebRTC Functionality](#webrtc-functionality)
   5. [Integration with External API Services](#integration-with-external-api-services)
      1. [AuthService](#authservice)
      2. [ChatService](#chatservice)
      3. [UserService](#userservice)
3. [Testing Locally](#testing-locally)
4. [Troubleshooting](#troubleshooting)

## **Prerequisites**

* .NET SDK (6.0 or later)
* Visual Studio Code or Visual Studio
* A browser (Chrome, Firefox, or Edge)
* Postman (for testing API endpoints)
* SQL Server 2019

## **Application Structure**

### Client-Side

Application Name: ChatApplication

Startup page : Login.html

* login.html: The login page where users authenticate through the **AuthService**.
* chat.html: The chat page where users send messages and start video calls using the **ChatService** and **UserService**.
* Profile.html: User profile management page with **UserService** integration.
* Register.html: For new user to sign up for the system through the **AuthService**.

Note : Api end points are specified in the javascript in each page.

### 2. Server-Side

1. **SignalRHub**: This SignalR Hub manages real-time communication (text and WebRTC).
2. **AuthService API**: Handles user authentication and session management.
3. **ChatService API**: Provides services to handle message storage and retrieval.
4. **UserService API**: Manages user profiles and information.

#### Server-Side: SignalR Setup

**SignalingService (C#)**

SignalRHub.cs: manages chat rooms, sending messages, and exchanging WebRTC data (SDP, ICE candidates).

**Program.cs**

Configures SignalR in the ASP.NET Core application.

Note: Allow the client application URL in the program.cs

#### Server-Side AuthService API:

Handles user login and authentication using an external API.

#### Server-Side ChatService API:

Manages messages, including sending and retrieving them via an external messaging API.

#### Server-Side UserService API:

Handles user profile information via an external API, allowing the client to manage user data.

## **Database Structure**

This section outlines how to set up the SQL Server database for the SignalR chat application, which includes tables for managing users and messages.

**Create Database and related tables**

* To begin, you'll need to create a new database named **chatSystem** for storing user and chat-related data.
* Once the chatSystem database is created, you'll need to define the following tables: Users for user management and Messages for storing chat messages.

Note: Database object file is attached to git

## **Testing Locally**

To test the chat application locally, follow these steps:

1. Making sure connection string is updated in appsettings.json file of API project and SignaR server-side project
2. Start all API projects: **AuthService**, **ChatService**, **UserService**.
3. Run the main ASP.NET Core project that serves the SignalR hub and the client application.
4. Open login.html in your browser (https://localhost:44326/login.html).
5. Create a new User using Sign Up page. Make sure user created in Users table.
6. Once logged in, test chat functionality, message saving, and video calls.
7. Open two browser window to test the multi user peer to peer communication.

## **Troubleshooting**

* **CORS Issues**: Ensure that your SignalR hub and API are correctly handling cross-origin requests.
* **Video Not Displaying**: Verify that the getUserMedia() permissions are correctly handled in your browser