Tech Stack Evaluation Document

Front-end framework

The front-end part of the app should aim to provide the following at minimum:

- A login page for existing users to sign in, or alternatively a sign-up page for new users to create their accounts
- An elegant, modern representation of a chat interface that provides all the necessary features like sending messages, selecting friends to send messages to, attaching files…

The range of front-end technologies is listed below, with their strength and weaknesses evaluated:

React

It is a component-based front-end Javascript framework. Its unique JSX syntax allows for the co-existence of HTML and JS code in the same file.

▼ Pros

- Reusable components
- Rich documentation and community
- Easy integration with other tech stacks like CSharp, ASP.NET

▼ Cons

- The learning curve for newcomers to JS
- Could be difficult to maintain as the project scales for the JS ecosystem is vast and not unified
- React uses JS, which is weakly typed, leading to poorly readable code (Could use TS which would address this problem)

Blazor

Blazor is a modern front-end web framework based on HTML, CSS, and C# that helps you build web apps faster. With Blazor, build web apps using reusable components that can be run from both the client and the server so that you can deliver great web experiences.

▼ Pros

- It offers a unified stack for both server-side and client-side development with C# and .NET.
- Seamless integration with the .NET ecosystem
- Offers reusable components to both the client and server-side

▼ Cons

- Might be incompatible with old browsers
- Smaller ecosystem compared to JS

Server-Client API

The major focus of any chat-related application is to maintain a real-time experience when it comes to sending and receiving messages, maintaining a stable and consistent connection between the messaging server and the communicating clients is a must in most circumstances. In a real-life scenario, we want to ensure that the delay between the sending and receiving of a message is within seconds at most.

There is a wide range of technologies/libraries that enables us to do that, a list of potential candidates is evaluated below using the classic strength-weakness analysis.

SignalR

It is an asp.net open-source library that enables real-time communications through **persistent connection** between the client and server. Which is different from the usual request-response model like in NodeJs.

▼ Pros

- Supports real-time functionalities - the ability to have the server push real-time content to connected clients in real-time, without needing a refresh button.

- Automatic fallback mechanisms
- SignalR can be scaled out across multiple servers, making it suitable for large-scale applications.

▼ Cons

- Limited native support for non-Windows platforms, though cross-platform support has improved in recent versions
- Learning curve for developers not familiar with .NET
- Potential performance overhead

▼ Alternatives

Azure pub-sub