Delhi Technological University

Department of Information Technology

Code: IT-303

Subject: Computer Networks

Topic: Chat-App Using WebSockets

Submitted by:

Rhythm Arya - 2K18/IT/094
Rishabh Jain - 2K18/IT/097

Introduction

The aim of this project is to create a real time messaging application. Unlike most chat applications within the market, this one can specialize in developers and can decide to boost its productivity, we will move onto the planning and technologies, on which we do our best to have everything ready for the development of the project. Afterwards, in the implementation section, we describe the most relevant bits of the development of the web application. Apart from real-time chat this application also includes bad word detection, will explain about this in more details in the later part of this documentation.

Project Idea

The key behind Real time chat application is web socket application through Client Server model. The WebSocket API is an approach to impart between a customer (client's program) and a worker. During the meeting, the information can stream bi-directional route continuously, which means the customer can send the messages to the server, and the server would response be able to back without the need to survey. Correspondence through the opened channel is durable and low idleness.

Client / Server Model is the basic of what makes communication through internet possible today. It is a long ways past the extent of this segment to make a comprehensive asset for client /server design. In any case, the essential administrators are indistinguishable.

This is intended to give a fundamental comprehension of the client /server model as indicated by LAN. For most web applications, correspondence conventions are not a topic of conversation.

AJAX through HTTP is the best approach since it is dependable and broadly upheld.

Notwithstanding, that isn't our case. We need, yet not in each and every circumstance, an amazingly quick specialized strategy to send/get messages progressively. For informing, there are a couple of correspondence conventions accessible for the web. The most well known ones are AJAX, Web Sockets, and WebRTC.
AJAX is a moderate methodology. Not just due to the headers that must be sent in each solicitation, yet additionally, and more significant, on the grounds that it is extremely unlikely to get advised of new messages in a visit room. By utilizing AJAX, we would need to ask for/pull new messages from the worker at regular intervals, which would bring about new messages to take up to a couple of moments to show up on the screen.

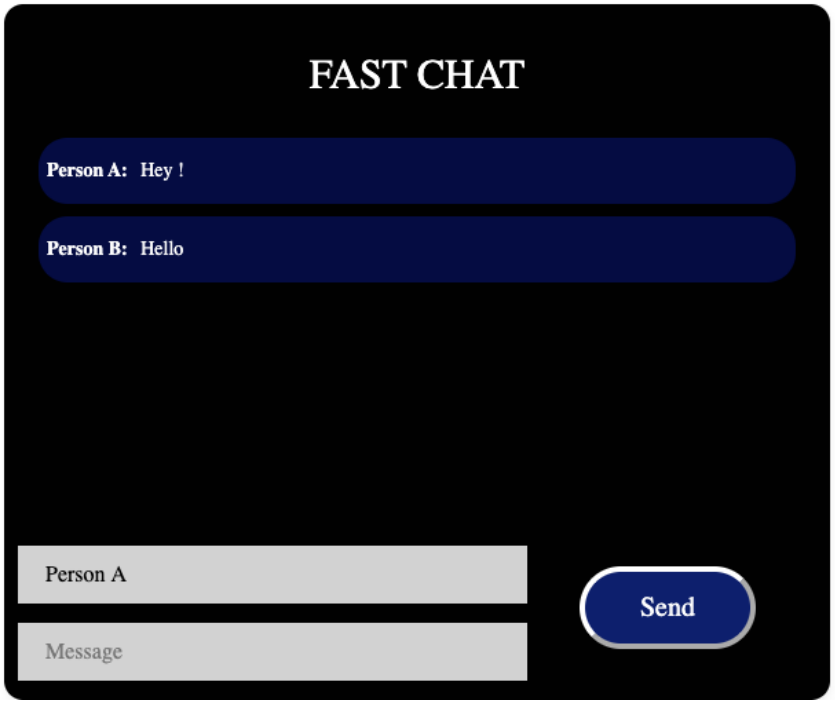
Web Sockets are a superior methodology. Web Sockets associations can take up to not many seconds to build up, however on account of the full-duplex correspondence channel, messages can be traded quickly (averaging not many milliseconds delay per message). Likewise, both customer and worker can get told of new demands through a similar correspondence channel, which implies that not at all like AJAX, the customer doesn't need to send the worker an appeal to recover new messages but instead trust that the worker will send them.

Project Features

- **Send and receive messages in Real-Time**

When a user connects to our application, he himself sets is username in the Handle input ,

For example “*Person A”*.

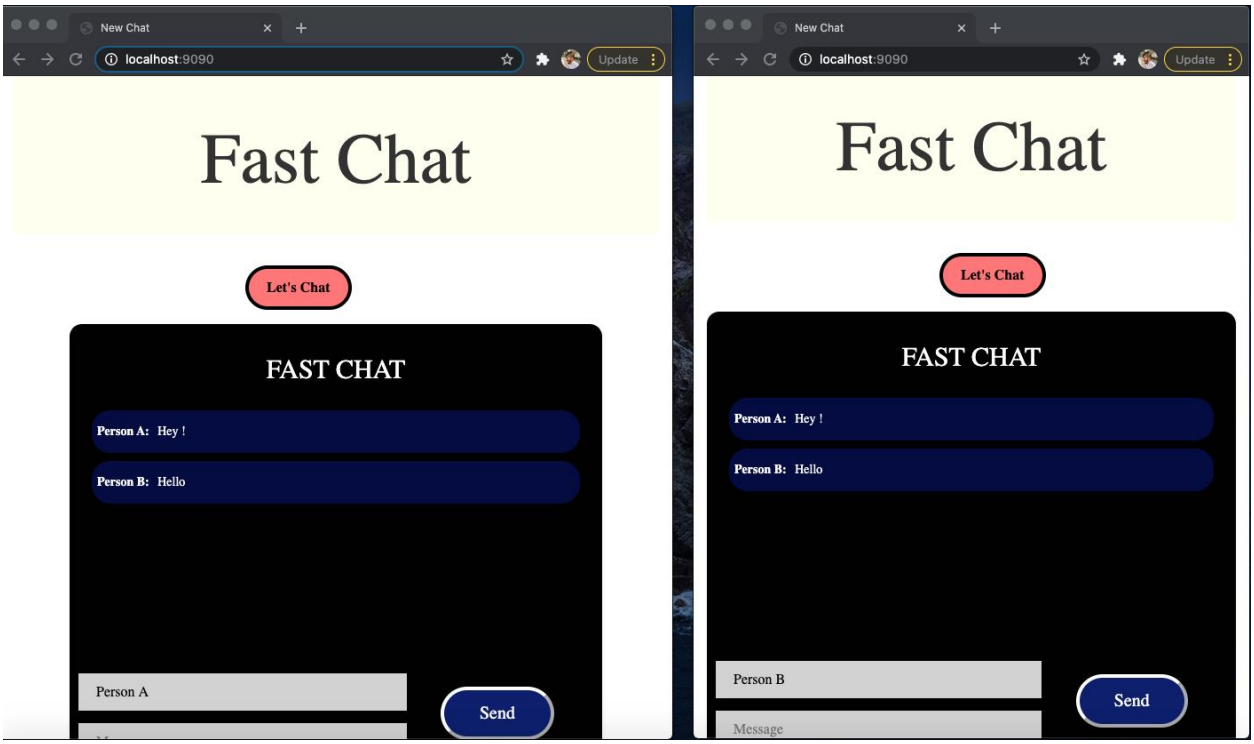


To do that we have to go on the server side (index.js) and add a key to the socket. Actually, sockets represent each client connected to our server.

On the client side, the goal is to do the opposite. Each time the button *change username* is clicked, the client will send an event with the new value.

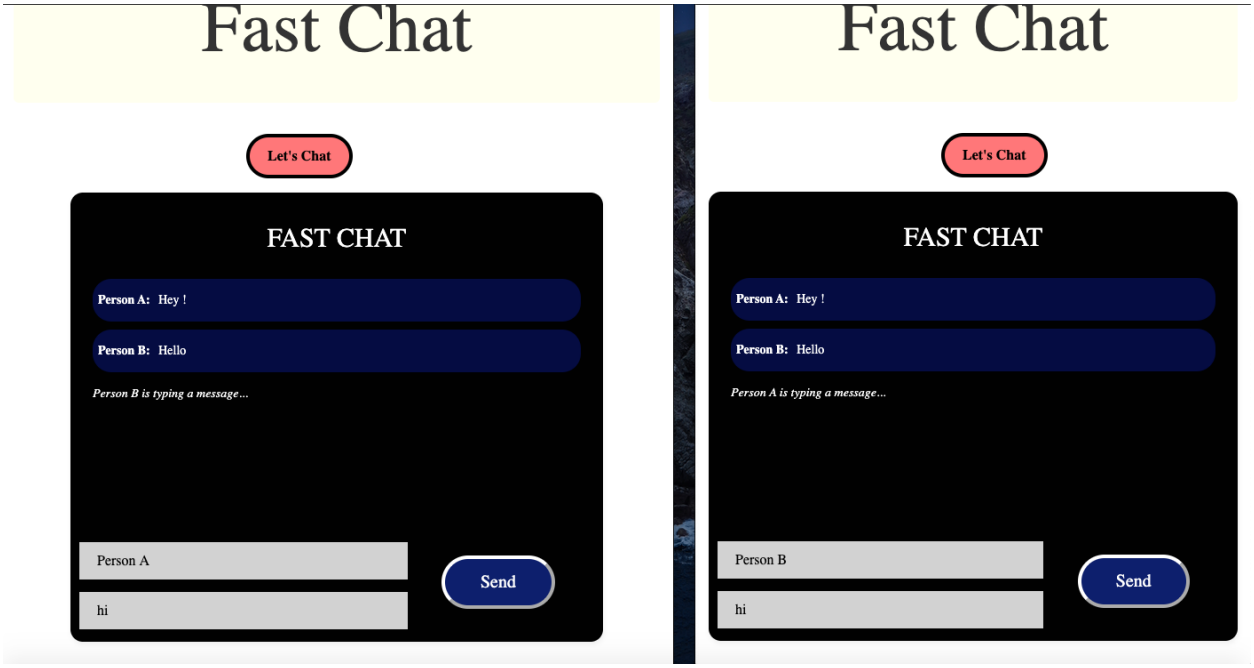
For the new message, you can see that we call the attachments property of io. It speaks to all the attachments associated. So this line will really make an impression on all the attachments. We need that to show a message sent by a client to all

Here is the final result of our chat app:



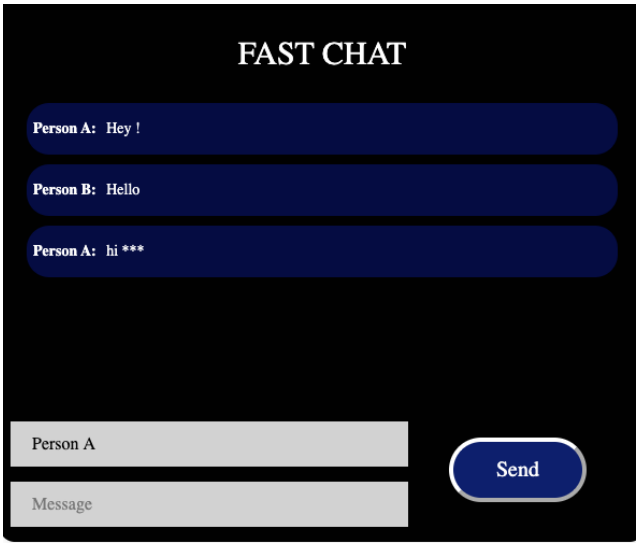
- **Broadcast a message.**

Broadcasting means sending a message to everyone else except for the socket that starts it.
We listen to *typing* and we broadcast a message.



As we can see when Person A is typing we could see a broadcast message “Person A is typing message” in the chat window of Person B and vice-versa.

- **Bad word Detection**



Apart from the above two features, our chat-app also includes a bad word detector, wherein i have created an array of some restricted words & if a client (person who is chatting) uses that word, then in place of that word “ *** ” will be sent so as to show that this word is a bad word.

Results

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications.

Future Work:

This is just a basic chat application but there are many more improvements you can make to it. To improve this app, we can add the following things:

- A registration system with the possibility to chat in a one-to-one chat room.
- History of all the conversations.
- Online/offline labels.
- Copy every feature of WhatsApp !

References:

1. https://medium.com/@noufel.gouirhate/build-a-simple-chat-app-with-node-js-and-socket-io-ea716c093088

2. https://medium.com/@JoshiRabindra/real-time-chat-app-using-websockets-part-2-38a89382d930

3. https://scotch.io/bar-talk/build-a-realtime-chat-server-with-go-and-websockets

4. https://dev.to/spukas/learn-websockets-by-building-simple-chat-app-dev

5. https://learntomato.flashrouters.com/what-is-a-client-what-is-a-server-what-is-a-host/