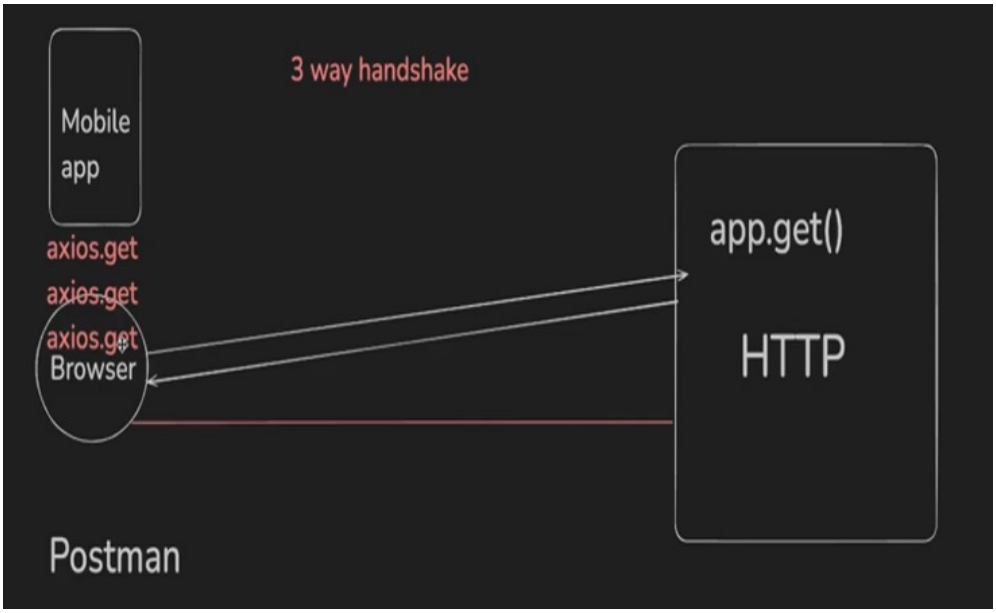
SocketIO is used by Canva and razerpay

Websocket protocol is like http only

WebSockets provide a way to establish a persistent, full-duplex communication channel over a single TCP connection between the client (typically a web browser) and the server.

Persistent = Its like when a http server send a req to backend then the frontend receives the answer but the connection breaks and if it req again any information then it has to again create a 3 way handshake and then get the http req is created and when response it received the connection is closed. ALSO CALLED AS REQUEST RESPONSE MODEL



To make a persistent connection we need **websockets** and **webrtc**

Full duplex = Its like both ways browser can send multiple messages and client can also receive and send multiple messages

TCP Connection(Transmition control protocol) /UDP =

http , websockets uses TCP under the hood and webrtc also can use TCP

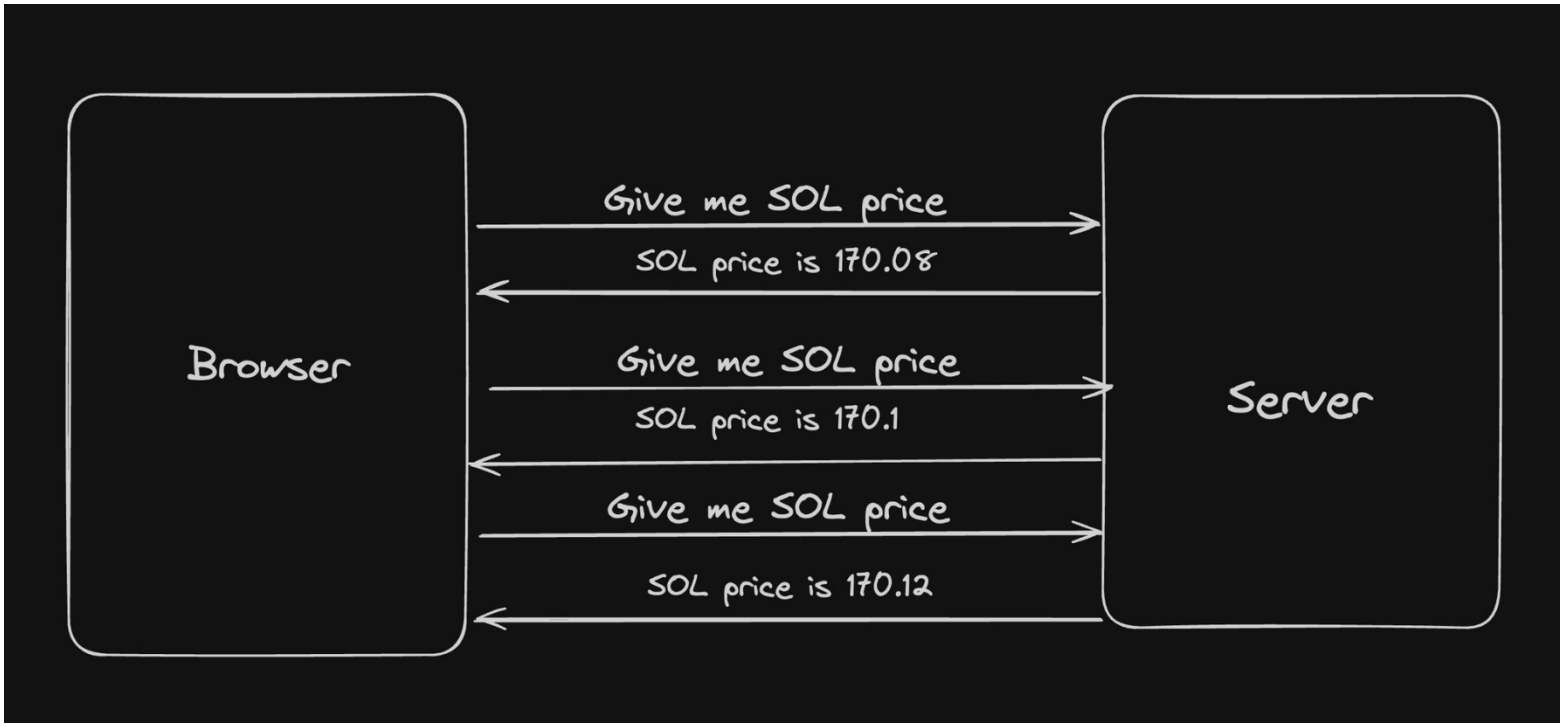
webrtc uses UDP under the hood

Why do we need Websockets?

For Real time communication (RTC) for applications like backpack.exchange here numbers changes in real time, metaverse game gather town

BUT FOR THIS THING WE CAN ALSO USE HTTP?

Yes but the issue is the number of req will increase because we need to send req multiple times in like 5ms to get the data and due to increase in number of req the load will also increase and also the 3 way connection will be increased.

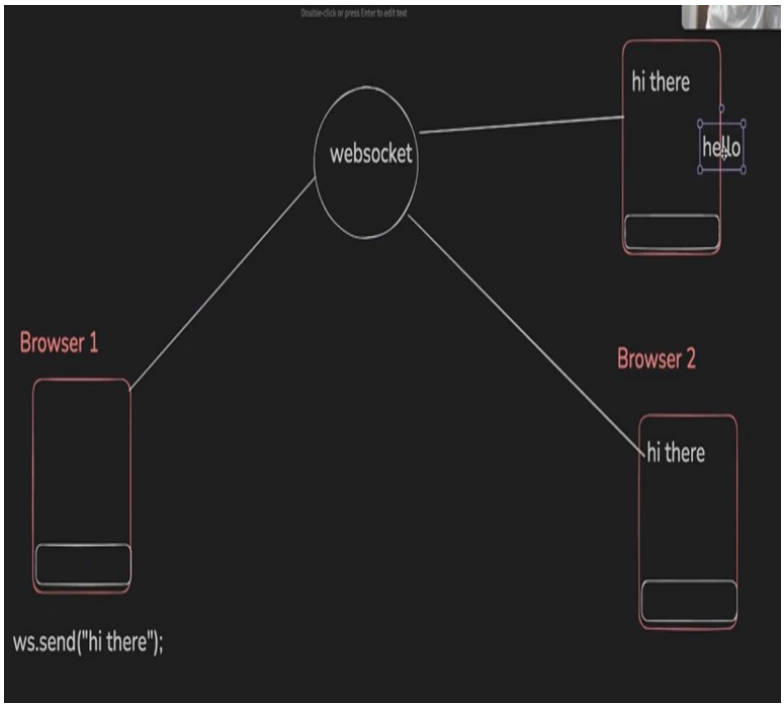


Use Cases for WebSockets:

- **Real-Time Applications**: Chat applications, live sports updates, real-time gaming, and any application requiring instant updates can benefit from WebSockets.
- **Live Feeds**: Financial tickers, news feeds, and social media updates are examples where WebSockets can be used to push live data to users.
- **Interactive Services**: Collaborative editing tools, live customer support chat, and interactive webinars can use WebSockets to enhance user interaction

Websockets is a 2 way communication its like server can also send data and client can also send/receive data.

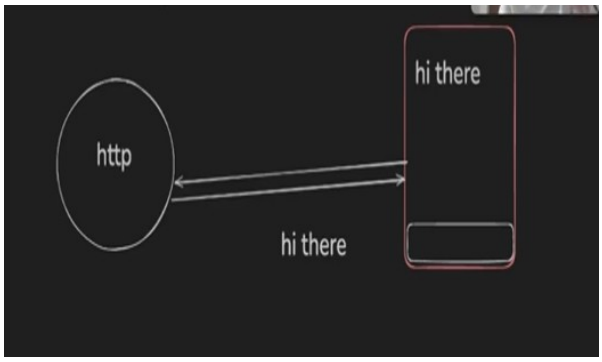
In websockets its easy to do



In this the msg can be send by any browser and received by any browser all are kind to connected

If chat application using HTTP?

But in http server the server cannot send the data as it it not a 2 way connection so to do that browser 2 has to request every 5 sec kuch aaya kya? Abh kuch aaya?



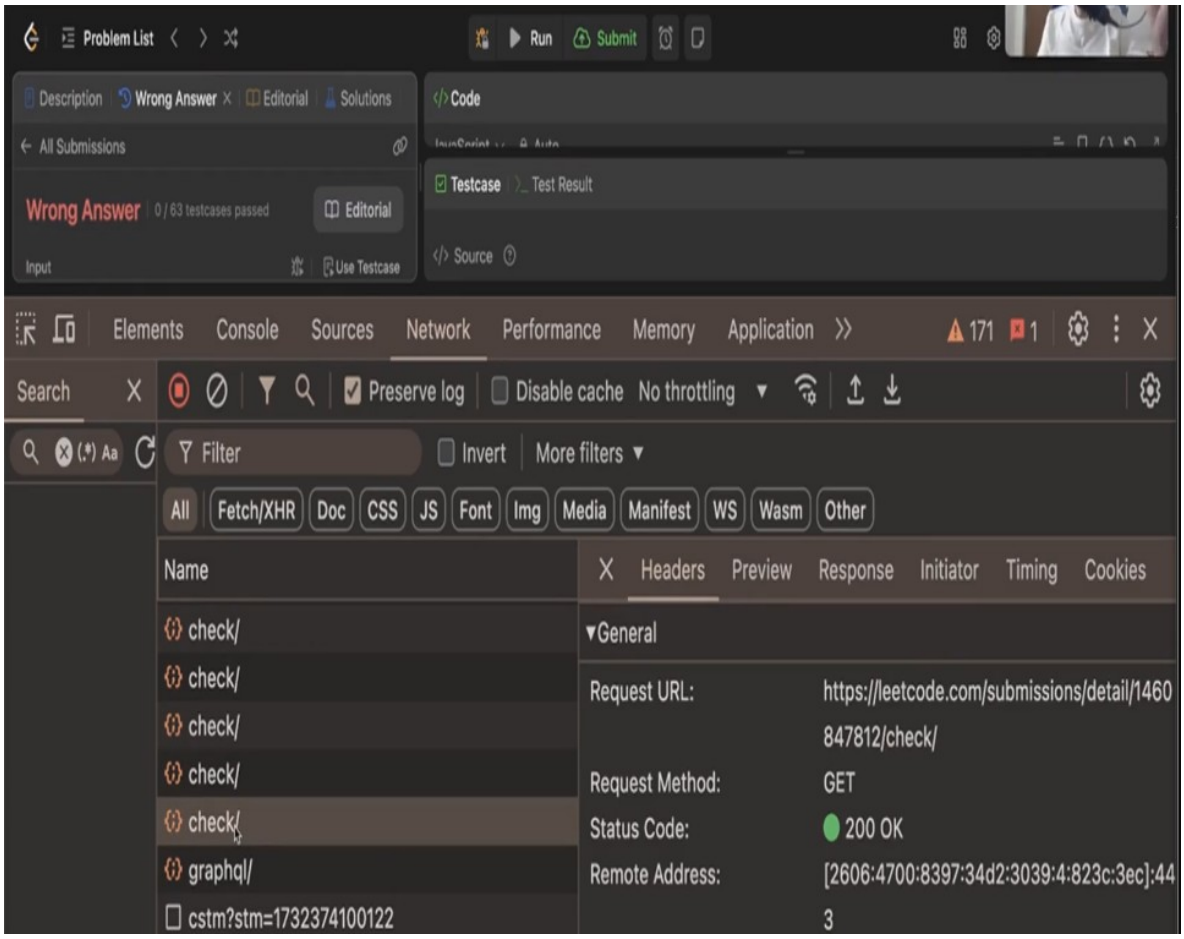
You need to use Polling/Long Polling

Polling means puchte rehna ki msg aaya kya ki nhi server sae

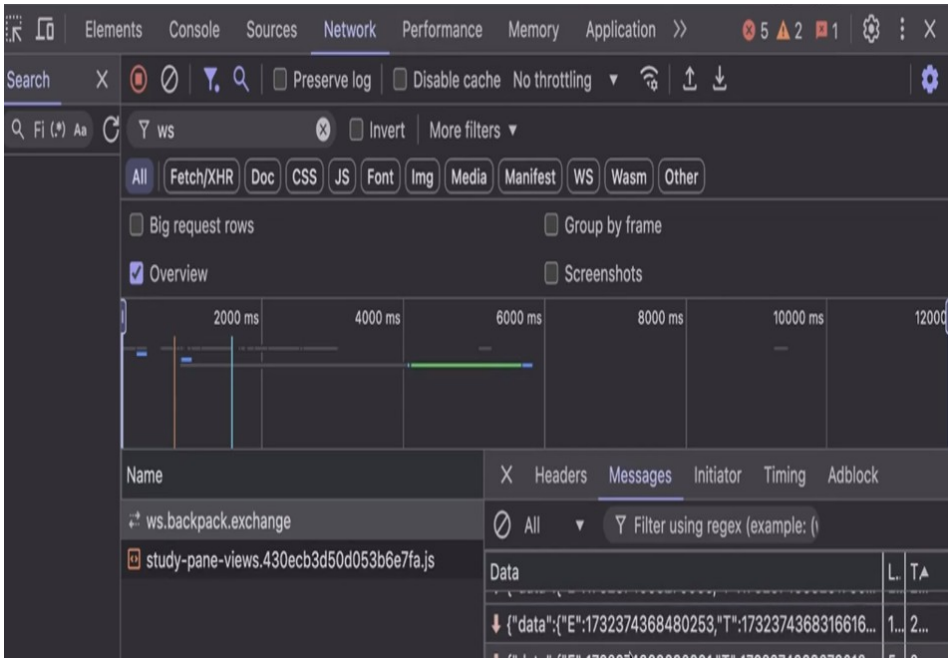
Can you create duplex application using http? YES USING POLLING

1. Network Handshake happens for every request
2. No way to push server side events (You can use polling but not the best approach)

LEETCODE USES HTTP POLLING



Websockets connection looks like this



There are inf messages going on between the server and client side

Setup the TS file and then

Using a library ws

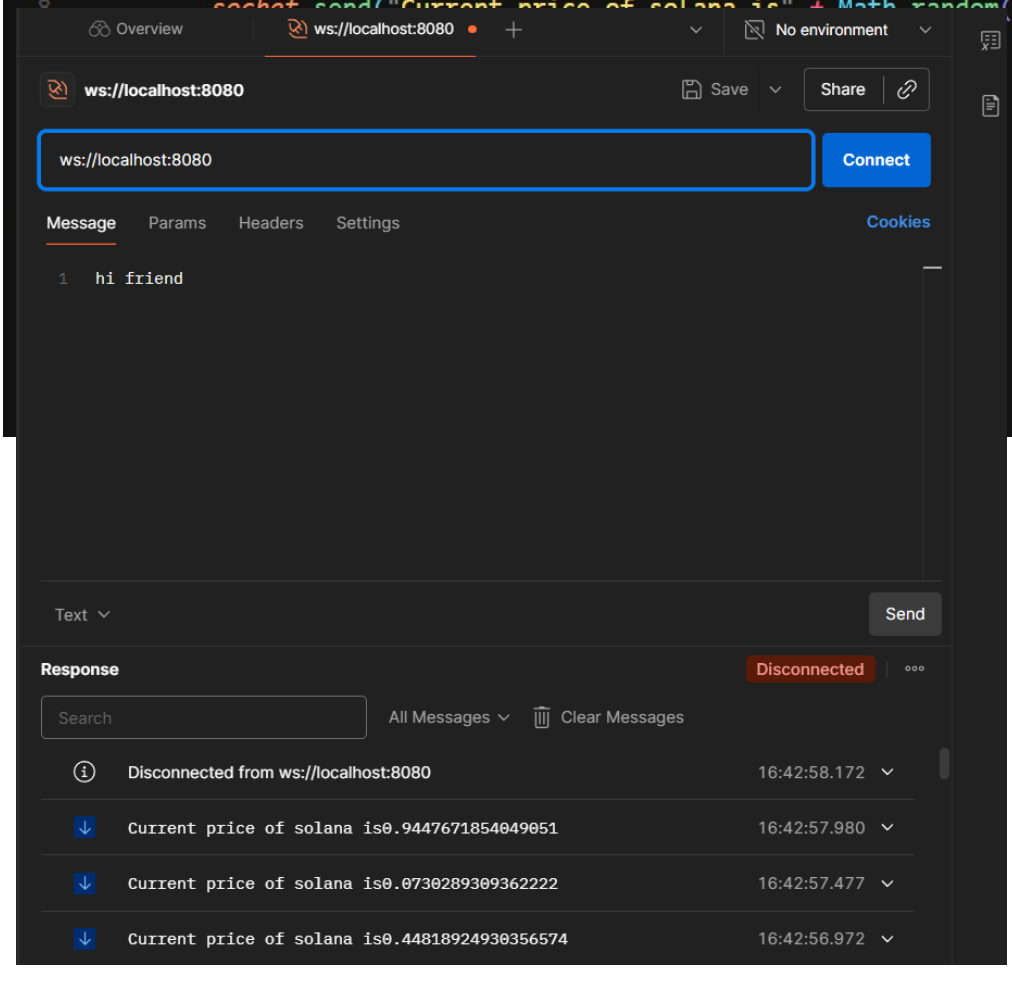
Why there is a bracket in { webSocketServer } because there are multiple things inside the server that’s why

```
wss.on("connection" ,function(socket){


})


app.get("/users", (req,res)=>{


})
```

This is same like http only

Server client code

```
import { WebSocketServer } from 'ws';
const wss = new WebSocketServer({port:8080});

//event handler
wss.on("connection" ,function(socket){
    console.log("user connected");
    setInterval(() => {
```



Interaction is done using postman and to add websockets go to the new option of the three lines and click on new to add and add the url

ws://localhost:8080

Echo application = client says hi and then server says hi

ping pong = client says ping and server says pong

```jsx
import { useEffect,useState,useRef } from 'react'
import './App.css'

function App() {
  //ws variable we need outside to send the msg
  const [ws,setWs] = useState();
  function sendMessage(){
    ws.send("ping");
  }
  //the below code will only run on mount
  useEffect(() =>{
    const ws = new WebSocket("ws://localhost:8080");
    setWs(ws);
    //receive a message
    ws.onmessage = (ev) =>{
      console.log(ev.data);
    }
  },[]);
  return (
    <div>
      <input type="text" placeholder="Message..."></input>
      <button onClick={sendMessage}>Send</button>
    </div>
  )
}

export default App
```

In this we have hard coded that any text you send the output will be pong only and use state is created to use the ws variable inside the sendmessage function

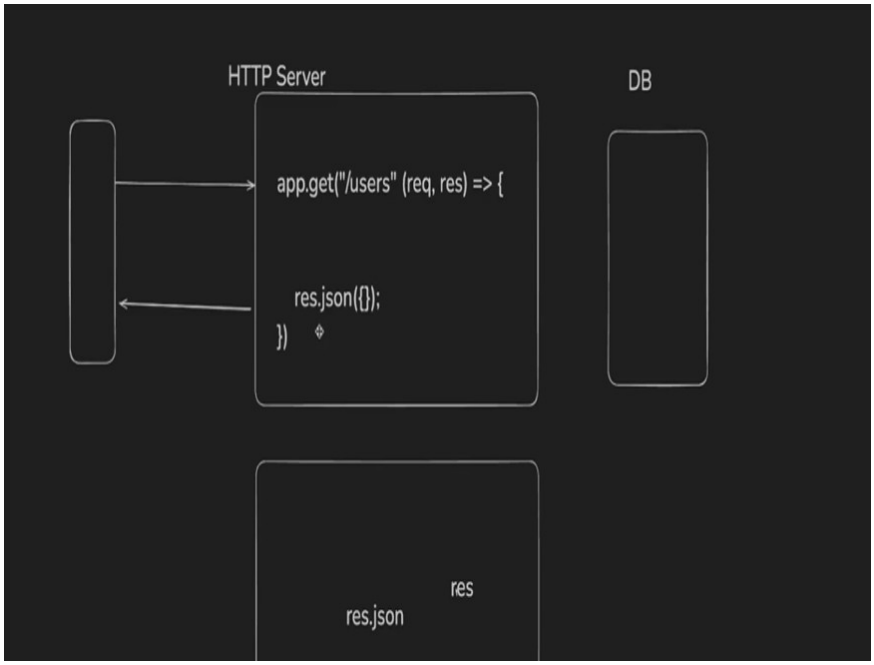
Scaling ws servers

http servers are stateless as all the users and orders are stored in the databases so during like normal session assume 2 http servers are used and during Diwali its like 6 and then after that the company can shuts down the extra servers without losing any data as the users array and orders array are stored in db

websockets can be stateless and cannot be, they are also sticky and stateful

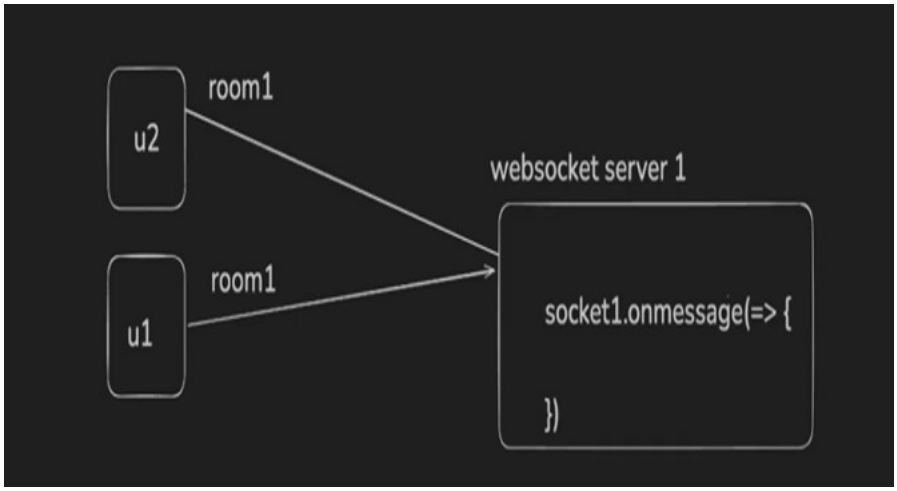
can we store sockets in a server?? YOU CANNOT DO THAT 😊

CAN WE STORE THE RES.JSON DATA IN DB? NOOOO….



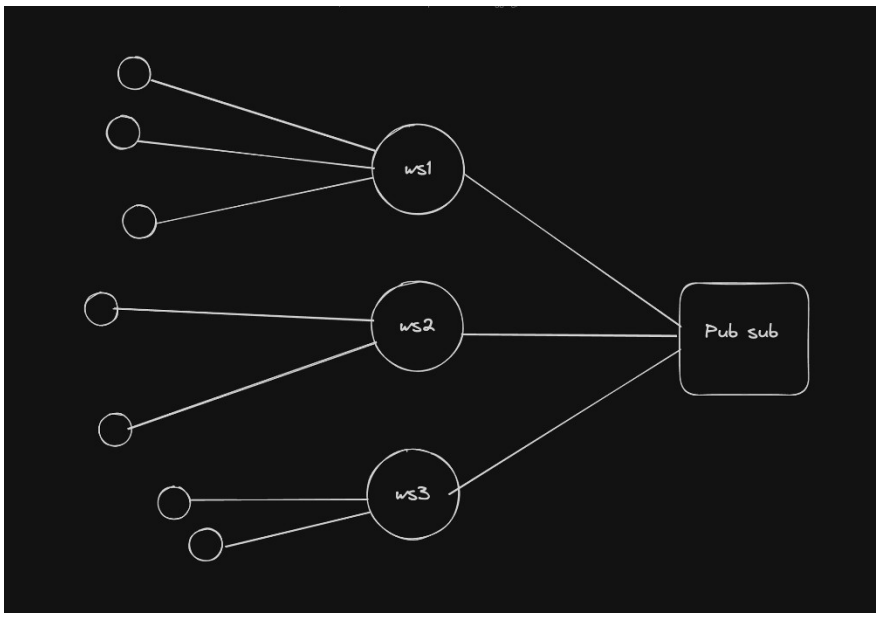
it is because the connection is just between the user 1 and the server 1 so the data cannot be stored and also db cannot store the string and all. Or you somehow stores and gives that data to server 2 to return to user that's also not possible because the user 1 is not connected to server 2 as it is connected to server 1 same for the sockets in the websockets.

THIS IS CALLED AS **STICKY CONNECTION**



Like if the user1,2 want to chat they must be in the server 1 not in server 2

That's why scalling up and scalling down is hard.



TO scale this you need PUBSUB (Scalable Websockets)

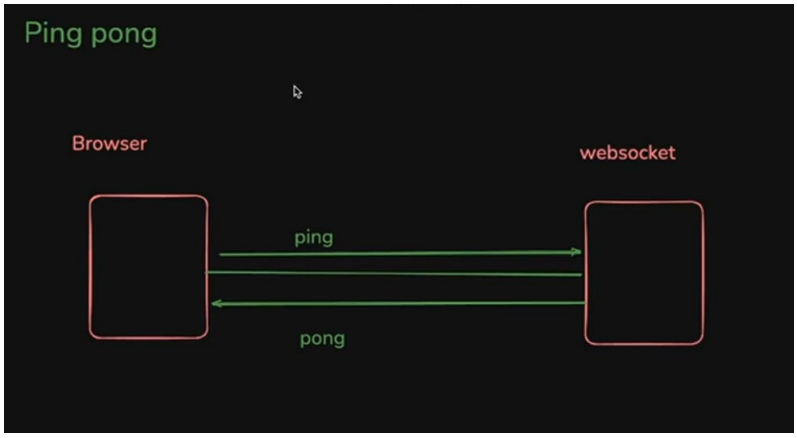
this is used to scale the server

imagine person from ws1 has to send a hi msg to everyone then it will be

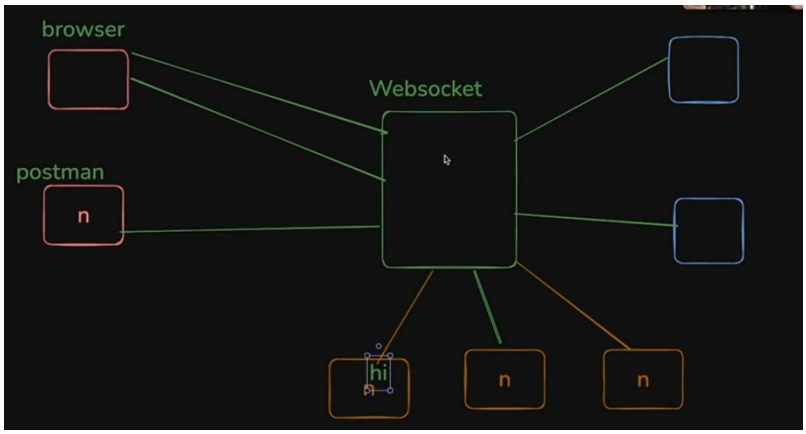
first send to ws1 and then to pubsub and then it can send to everyone this is how it can be done to decrease the load.

Pubsub is like a central things which helps us to reduce the load of just using one websocket server and this helps to connect the user to different servers but with same team code also

What we made is a ping pong application



Polling is also called as heartbeating



What are we making is a websocket server in which every person has its own team code to join if a chat is sent in room 3 it must send to the chat room 3 only not to the other teams so this is what we are building

Simple broadcast chat application

```
1  import { WebSocketServer } from "ws";
2
3  const wss = new WebSocketServer({port :8080})
4
5  import express from "express"
6  const app = express()
7
8  app.listen(3000 )
```

the above 2 lines define a websocket server and start

the below 3 lines define a express server and start

```
wss.on("connection", (socket) => {

})


//
app.post("/signup", (req, res) => {
    ....
})
```

whenever someone sends a request in the http server it calls the callback function if you want to respond back use the res object if you want to get the data you use the req object then

req=> input mai kya aa rha hai

res=> to send back messages

similarly in the websocket server whenever someone connects to the ws server the function will be called with a reference to a socket

```
import { WebSocketServer } from "ws";

const wss = new WebSocketServer({port :8080})

let userCount = 0;

wss.on("connection" , (socket)=> {
    userCount+=1;
    console.log("user connected #" + userCount);

})
```

This will show user connected whenever a connection is made using postman and will log it

```
import { WebSocketServer } from "ws";
const wss = new WebSocketServer({port :8080})
let userCount = 0;

wss.on("connection" , (socket)=> {
   userCount+=1;
   console.log("user connected #" + userCount);
   socket.on("message",(message)=>{
       console.log("message received "+message.toString())
       setTimeout(()=>{
           socket.send(message.toString() + ": sent from the server")
       },1000)
   })
})
```

the socket.on is used to show the message in the console its like app.get in the http server

in this the server receive the msg and sends the exact message to server again using the socket.send after 1 second

THIS IS A PING PONG APPLICATION THO HEHEHEHEH

Another variant of postman was postwoman which later changed into **hoppscotch**

Using this to like use it as a second user

```
import { WebSocketServer,WebSocket } from "ws";
const wss = new WebSocketServer({port :8080})
let userCount = 0;
let allSockets:WebSocket[] = [];

wss.on("connection" , (socket)=> {
   allSockets.push(socket);

   userCount+=1;
   console.log("user connected #" + userCount);
   socket.on("message",(message)=>{
       console.log("message received "+message.toString())
       for (let i=0;i<allSockets.length;i++){
           const s = allSockets[i]!;
           s.send(message.toString() + ": sent from the server")
       }
   })
})
```

Now the issue is if we send a message from postwoman it stays with the postwoman only the postman cannot view that msg means how can we bring a broadcast feature

A chat application with broadcast feature

in this new thing I learnt is if it says s is not defined just add a ! at the last to tell the other thing that its not undefined

This is a chat based app code

```
socket.on("message", (message) => {
    console.log("message received " + message.toString())
    allSockets.forEach(s => {
        s.send(message.toString() + ": sent from the server");
    })
})
```

Other way to write the for loop using foreach

But there is a issue if someone disconnects form the ws server the code will still send the data to the person bcoz its still present in the users array so we should remove that

```
16      socket.on("disconnect",()=>{
17          allSockets = allSockets.filter(x => x !=socket)
18      })
19  })
```

Delete the disconnected user

NOW CREATING THE COMPLICATED PART WHICH IS THE TEAMCODE THING

Create the Schema of your websocket server

Chat app:

- Join a room

  {
  “type”: “join”,
  “payload”:{
  “roomId”: ”123”,
  }
  }

-
```
{
  "type": "chat",
  "payload": {
    "message: "hi there"
  }
}
```

What the server can send/user can receive

-
```
▼ Message

{
  "type": "chat",
  "payload": {
    "message": "hi
  }
}
```

```
let allSockets ={
    "room1" :{socket1,socket2},
    "12212" : {socket},
    "12223312harikirat": {socket1, socket2, socket3}
};
```

This the changed arrow will look like after the ids are connected using team code

```
interface User {
    socket: WebSocket;
    room: string;
}

let allSockets: User[] = [];

[
    {socket: socket, room: "room1"},
    {socket: socket2, room: "room2"},
    {socket: socket3, room: "room1"},
]
```

Another way to storing the data in the user array using the interface

TS CAN USE SEND JUST STRING NOT THE JSON, OR ANY ANOTHER AND THE CODE WILL ALSO RECEIVE THE STRING ONLY

THE SCHEMA DATA WILL BE RECEIVED IN THE FORM OF STRING ONLY SO WE HAVE TO SOMEHOW CONVERT THAT INTO OBJECT

```
> let obj = {
    "name": "harkirat"
  }
< undefined
> let str = "{"name": "harkirat"}"
< undefined
> typeof obj
< 'object'
> typeof str
< 'string'
> obj.name
< 'harkirat'
> str.name
> let str2 = JSON.stringify(obj)
< undefined
> str2
< '{"name":"harkirat"}'
> JSON.parse(str2)
< ▶ {name: 'harkirat'}
```

WHY TO CONVERT INTO OBJECT?

Bcoz the in object we can do obj.name but not in the string

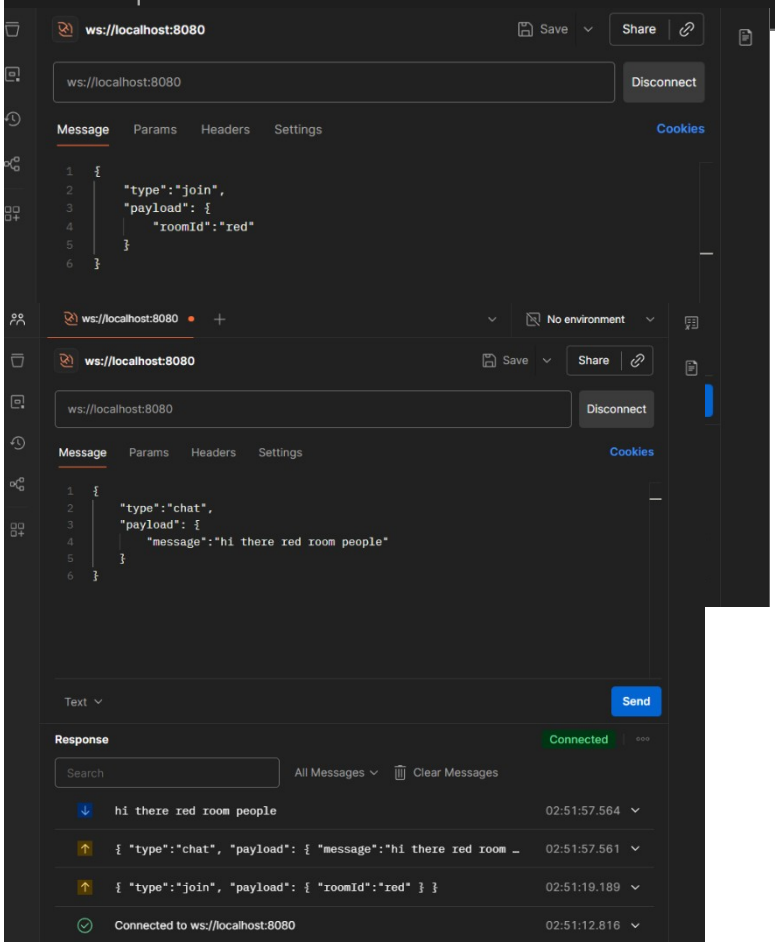
Convert the string into object using json.parse

```ts
import { WebSocketServer,WebSocket } from "ws";
const wss = new WebSocketServer({port :8080})

interface User{
   socket: WebSocket;
   room:string;
}
let allSockets: User[] = [];

wss.on("connection" , (socket)=> {
   socket.on("message",(message)=>{
       // @ts-ignore
       const parsedMessage = JSON.parse(message as unknown as string);
       if(parsedMessage.type === "join"){ // based on the schema
           allSockets.push({
               socket,
               room:parsedMessage.payload.roomId
           })
       }
       if(parsedMessage.type == "chat"){
           let currentUserRoom = null;
           for(let i=0;i<allSockets.length;i++){
               if(allSockets[i]?.socket == socket){
                   currentUserRoom = allSockets[i]?.room
               }
           }
           for(let i=0;i<allSockets.length;i++){
               if(allSockets[i]?.room == currentUserRoom){
                   allSockets[i]?.socket.send(parsedMessage.payload.message)
               }
           }

       }
   })
```

In the chat code we are trying to find the user's room and then iterating again to find which user has that room and sending data to that

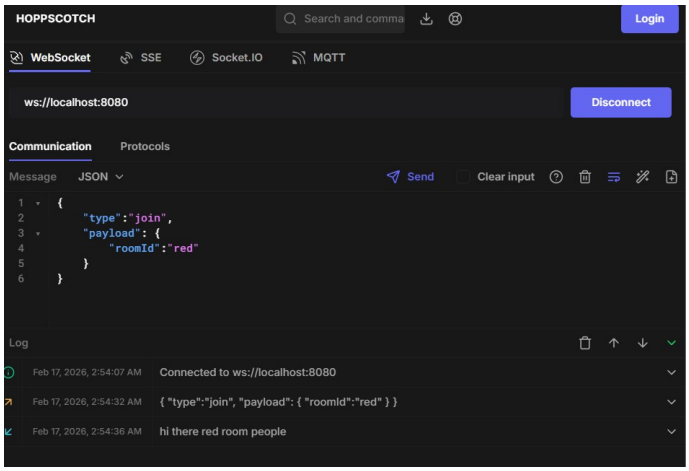
Message as unknown as string is just used to define that the message is not empty

Its like firstly identify that the person is part of the room or not if yes then send it to the person with that room not to any other room



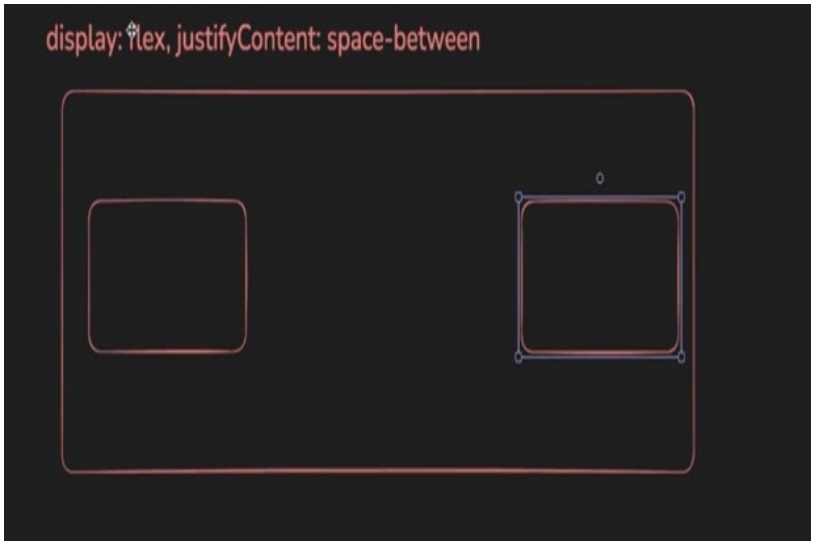
First connected it using postman and postwoman and then send the join schema and now I will send the message schema

Received the msg also means the code is working as usual



Received in the postwoman also as it was also in the red team

FRONTEND TIME




```
const [messages,setMessages] = useState(["hi there","hello"]);
useEffect(()=>{
    const ws = new WebSocket("http://localhost:3000")
    ws.onmessage = (event)=>{
        setMessages(m=> [...m,event.data])
    }
},[])
```

the …m part means it will take the array intial array with all the elements and later after that add the event.data into the array also to that

PUBSUB IS USED TO CREATE THE FAN OUT ARCHITECTURE AND THIS LIKE REDUCING THE PRESSURE OF USERS LIKE 1MIL TO SCALE WE CAN DO IT BY ADDING MORE AND MORE LAYERS