BROWSER CHAT

GROUP NO. 6

TEAM MEMBERS

- D.V.Harsha Vardhan Reddy
- Nihar Ranjan Murudi
- Pranjal Mishra
- Sumanth
- Yash Verma

STUDIO SHODWE

Features of Browser Chat

01 **Real time messaging**

02 **User status**

03 **Group Chat**

04 **Emojis feature**

Features of Browser Chat

Real time messaging

Provides real-time messaging capability by enabling users to send and receive messages instantly without delay

Group Chat

It allows users to chat with multiple users to join the conversation. This feature is useful for community discussions, and socializing.

Features of Browser Chat

User Statuses

It displays user status, indicating if a user is online or offline; if they are available for chat or busy with other tasks.

Emojis

Emojis feature add a fun element to chats and help to express emotions and feelings without using words.

APPLICATION PROTOCOL

Web Socket Protocol

WebSocket is a full duplex communication protocol that enables two-way communication between a client and a server over a single, long-lived connection. It provides a reliable and efficient way to send and receive messages in real-time.

APPLICATION PROTOCOL

1. **HTTP:** The code creates an HTTP server using the **http module** and uses it to listen for incoming requests.
2. **WebSocket:** The code uses the **socket.io module** to establish WebSocket connections with clients.

Transport Layer protocol

TCP

TCP is a reliable and connection-oriented protocol that provides a guaranteed delivery of data, error detection, and correction. It is used to establish a connection between the client and server and to manage the reliable delivery of messages.

Transport Layer protocol

TCP

The code uses the **net module** indirectly through the socket.io module, which relies on TCP/IP to establish and maintain WebSocket connections.

Network Layer protocol

IP

IP is a network layer protocol that provides the routing of data packets between hosts on different networks. It is used by TCP/IP to determine the most efficient route for data packets to travel between hosts

IP: The code specifies an IP address (10.7.26.121) to bind the server to.

CODE AND RESULT

```
var submitButton = document.getElementById("submitButton");
var sendMessage = document.getElementById("send");
var chatInput = document.getElementById("chatInput");

var name ='';
var socket = io();

//Event Listener
submitButton.addEventListener("click", login);
sendMessage.addEventListener('click',sendMessages);
chatInput.addEventListener('keypress', function (e) {
    if (e.key === 'Enter') {
        sendMessages();
    }
});

function sendMessages(){
    var chatInputValue = document.getElementById("chatInput").value;

    if(chatInputValue == ""){
        alert("Message can't be empty!")
    }else{

       socket.emit("new message",{ name:name, message:chatInputValue});
       document.getElementById("chatInput").value="";
    }
}
```

```
function uploadFile() {
    const fileInput = document.getElementById('fileUploadInput');
    const file = fileInput.files[0];

    const xhr = new XMLHttpRequest();
    const url = 'http://example.com/upload';

    xhr.onreadystatechange = function() {
      if (xhr.readyState === 4 && xhr.status === 200) {
        console.log('File uploaded successfully');
      }
    }

    xhr.open('POST', url, true);

    const formData = new FormData();
    formData.append('file', file);

    xhr.send(formData);
  }
```

CODE AND RESULT

```
socket.on("new message", function (data) {
    //console.log(user);
    updateMessageList(data);
 })

 function updateMessageList(data){
    var messageList = document.getElementById("messageList");
        messageList.appendChild(createMessageList(data));
}


function createMessageList(data) {
    console.log(data);
    let li = document.createElement('li');
    let span = document.createElement('span');
    span.textContent = data.name;
    li.textContent = data.message;
    li.appendChild(span);
    return li;
}


socket.on("has connected", function(users){
    updateUserList(users);

})

socket.on("has disconnected",function (users) {
    updateUserList(users)
 })
```

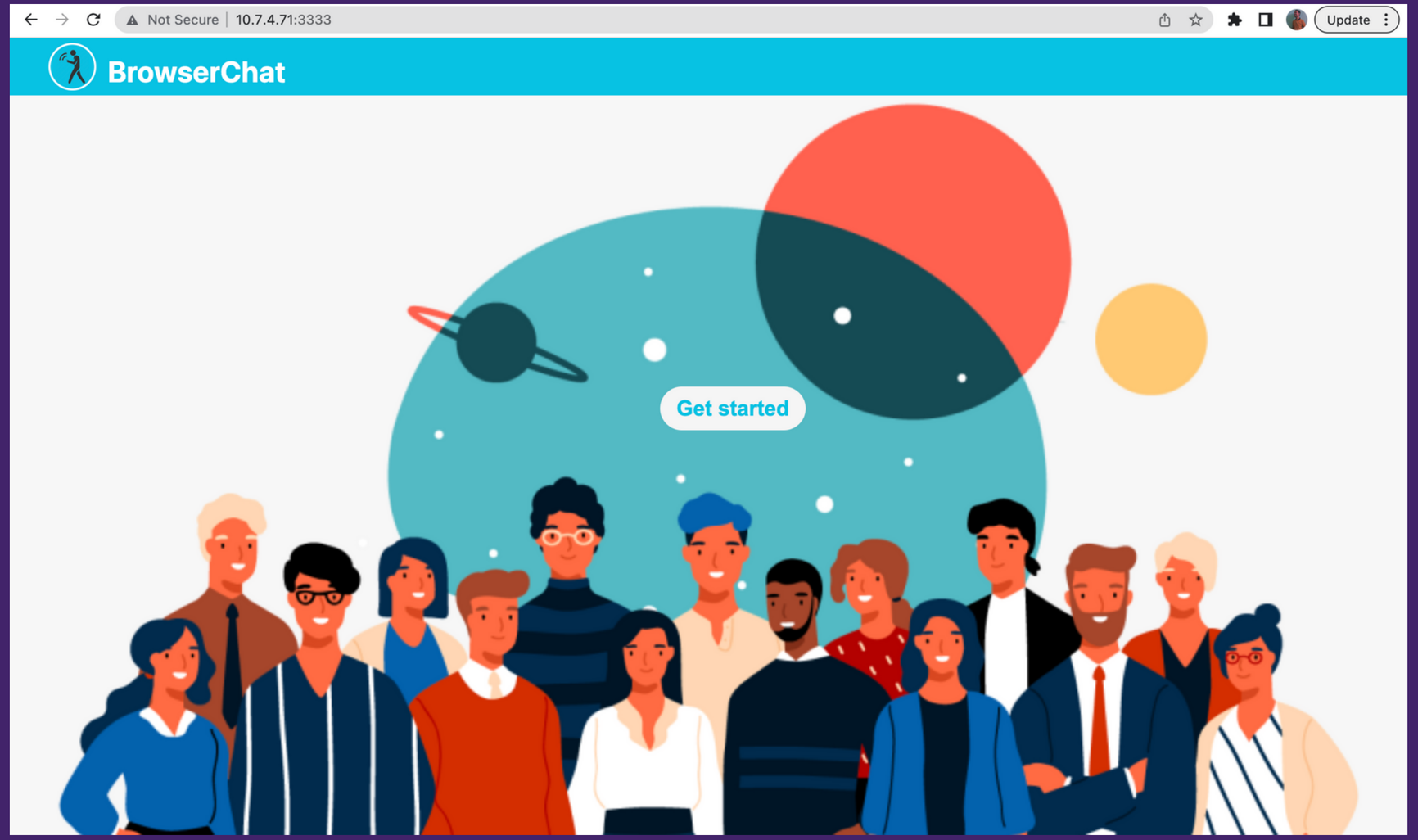
```
function updateUserList(users){
    var list = document.getElementById("list");
    list.innerHTML="";
    for(var i=0;i<users.length;i++){
        list.appendChild(createList(`${users[i]}`));
    }
}


function createList(name) {
    let li = document.createElement('li');
    li.textContent = name;
    return li;
}

function login(e) {
    e.preventDefault();
    var submitButtonValue = document.getElementById('username').value;
    if(submitButtonValue == ''){
        document.getElementById('warningMessage').style.display="block";
        document.getElementById('warningMessage').textContent="* Username can't be empty.";

    }else{
        var usernameRegex = /^[a-zA-Z0-9]+$/;
        var validfirstUsername = submitButtonValue.match(usernameRegex);
        if(validfirstUsername == null){
            document.getElementById('warningMessage').style.display="block";
        document.getElementById('warningMessage').textContent="* Only characters A-Z, a-z are  acceptable.";
        }else{
        name = submitButtonValue;
        document.getElementById('loginForm').style.display="none";
        document.getElementById('chatRoom').style.display="block";
        socket.emit("has connected",submitButtonValue);
        document.getElementById("messageList").innerHTML="";
```

CODE AND RESULT

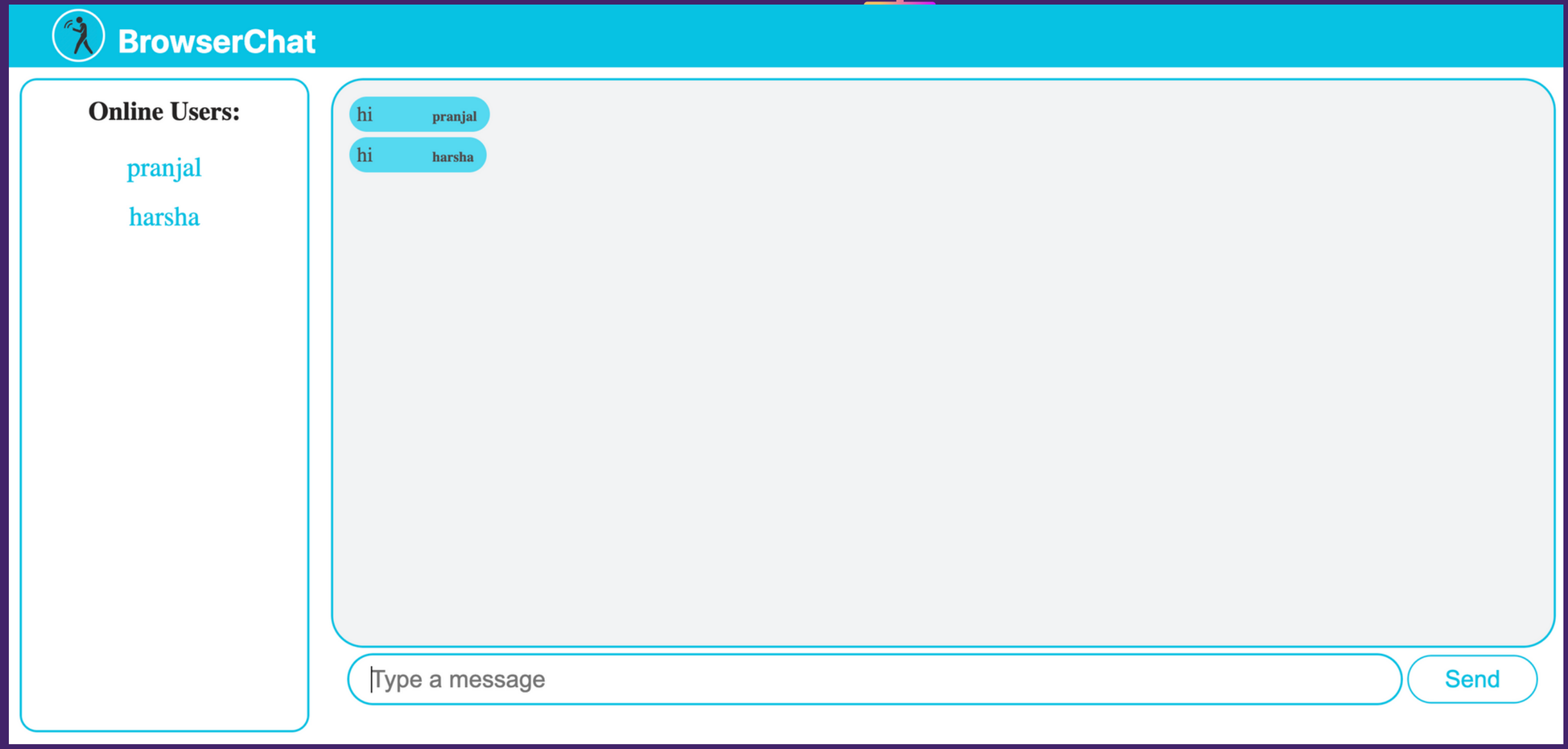
```
var express = require('express');
var http = require('http');
const { isIPv4 } = require('net');
var socket = require('socket.io');
var users = [];

var app = express();
var server = http.Server(app);
var io = socket(server);
const ipaddress ='127.0.0.1';
const PORT = process.env.PORT || 3333;
server.listen(PORT, () => console.log(`Server is up and running on ${ipaddress}:${PORT}`));


app.get("/",function(req,res){
    res.sendFile(__dirname + "/index.html");
});

app.get("/styles/style.css",function(req,res){
    res.sendFile(__dirname + "/styles/style.css");
});

app.get("/js/script.js",function(req,res){
    res.sendFile(__dirname + "/js/script.js");
});
```

```
app.get("/img/logo.png",function(req,res){
    res.sendFile(__dirname + "/img/logo.png");
});


io.on("connection", function (socket) {
    console.log("A user has connected!");
     var name ="";
    socket.on("has connected",function(username){
      name = username;
      users.push(name);
      io.emit("has connected",users);
    })

    socket.on("disconnect", function(){
       users.splice(users.indexOf(name),1);
       io.emit("has disconnected",users);
    });

    socket.on("new message",function(data){
       io.emit("new message",data);
    });
})
```

CODE AND RESULT



CODE AND RESULT



Github Link

https://github.com/HARSHABMU/DC-CN-BROWSER-CHAT-PROJECT

THANK YOU