COMPUTER NETWORKS PROJECT REPORT
(IT - 303)
REAL TIME CHAT APP WITH NODE.JS AND SOCKET.IO

BY : TEENA SINGH (2K18/IT/126)
UTKARSH KHARB(2K18/IT/128)

SUBMITTED TO:
MRS ANAMIKA CHOUHAN

INDEX

ACKNOWLEDGEMENT

I would like to take this opportunity to express my profound gratitude and deep regards to my teacher, **Professor Anamika Chouhan**, for her exemplary guidance, monitoring and constant encouragement through the course of this report.

The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life in which I am about to embark.

INTRODUCTION

In our project we are going to use the Node.js and socket.io platform to create a real-time chat application that instantly sends and displays messages to a recipient without refreshing the tab. To accomplish this, we will be using the JavaScript application and the Socket.io libraries.

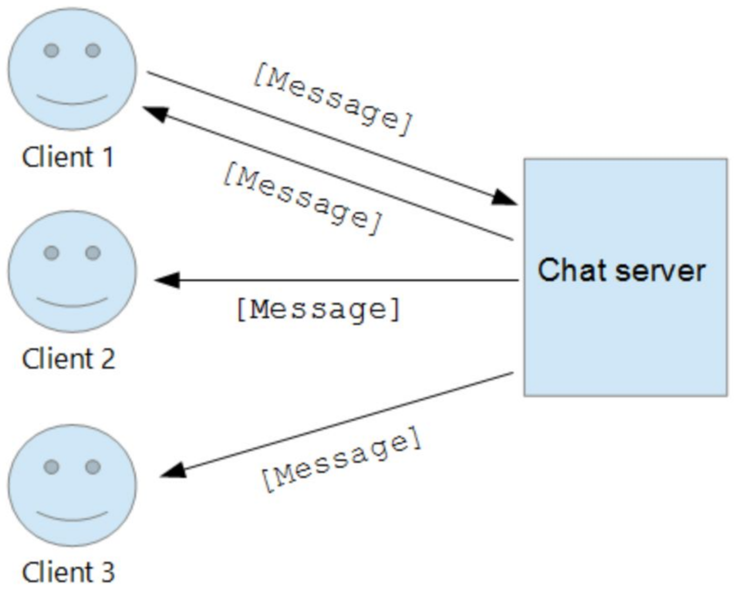
Teleconferencing or Chatting, is a method of using technology to bring people and ideas“together” despite geographical barriers. The technology has been available for years but he accepted it recently. Our project is an example of a chat server. It is made up of 2applications the client application, which runs on the user’s Pc and server application, which runs on any Pc on the network. To start chatting clients should get connected to a server where they can practice two kinds of chatting, public one (message is broadcasted to all connected users) and private one (between any 2 users only) and during the last one security measures were taken.

AIM

Our aim is to create a real time chat application complete with usernames and connect/disconnect messages. We will use Socket.io to manage real time web socket connections to a Node.js server that will allow you to communicate real time chat messages to all clients connected to a single server.

This entire project will take less than 100 lines of JavaScript and almost no HTML/CSS, but it is incredibly powerful what can be done with such little code.

IMPLEMENTATION



Node.js

Node.js is a cross-platform, open-source JavaScript runtime framework running JavaScript code outside the browser. The most significant benefit of using Node is that we can use JavaScript as a language that is both front-end and back-end.

As we know, JavaScript was mainly used for client-side scripting in which scripts were inserted in the HTML of a web page and run client-side in the web browser of the user by a JavaScript engine.

Node.js helps developers to write Command Line tools using JavaScript and server-side scripting to run server-side scripts to create dynamic web page content before the page is submitted to the web browser of the user.

Socket.io

Socket.IO is a JavaScript library for realtime web applications. It enables real time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Socket.io enables real-time bidirectional event-based communication.

A socket is an object that represents a low level access point to the IP stack. This socket can be opened or closed or one of a set number of intermediate states. A socket can send and receive data down disconnection. Data is generally sent in blocks of few kilobytes at a time for efficiency; each of these blocks are called a packet.All packets that travel on the internet must use the Internet Protocol. This means that the source IP address, destination address must be included in the packet. Most packets also contain port numbers. A port is simply a number between 1 and 65,535 that is used to differentiate higher protocols. Ports are important when it comes to programming your own network applications because no two applications can use the same port.Packets that contain port numbers come in two flavors: UDP and TCP/IP. UDP has lower latency than TCP/IP, especially on startup. Where data integrity is not of the utmost concern, UDP can prove easier to use than TCP, but it should never be used where data integrity is more important than performance; however, data sent by UDP can sometimes arrive in the wrong order and be effectively useless to the receiver. TCP/IP is more complex than UDP and has generally longer latencies, but it does guarantee that data does not become corrupted when travelling over the internet. TCP is ideal for file transfer, where a corrupt file is more unacceptable than a slow download; however, it is unsuited to internet radio, where the odd sound out of place is more acceptable than long gaps of silence.

SYSTEM REQUIREMENTS

Software Requirements

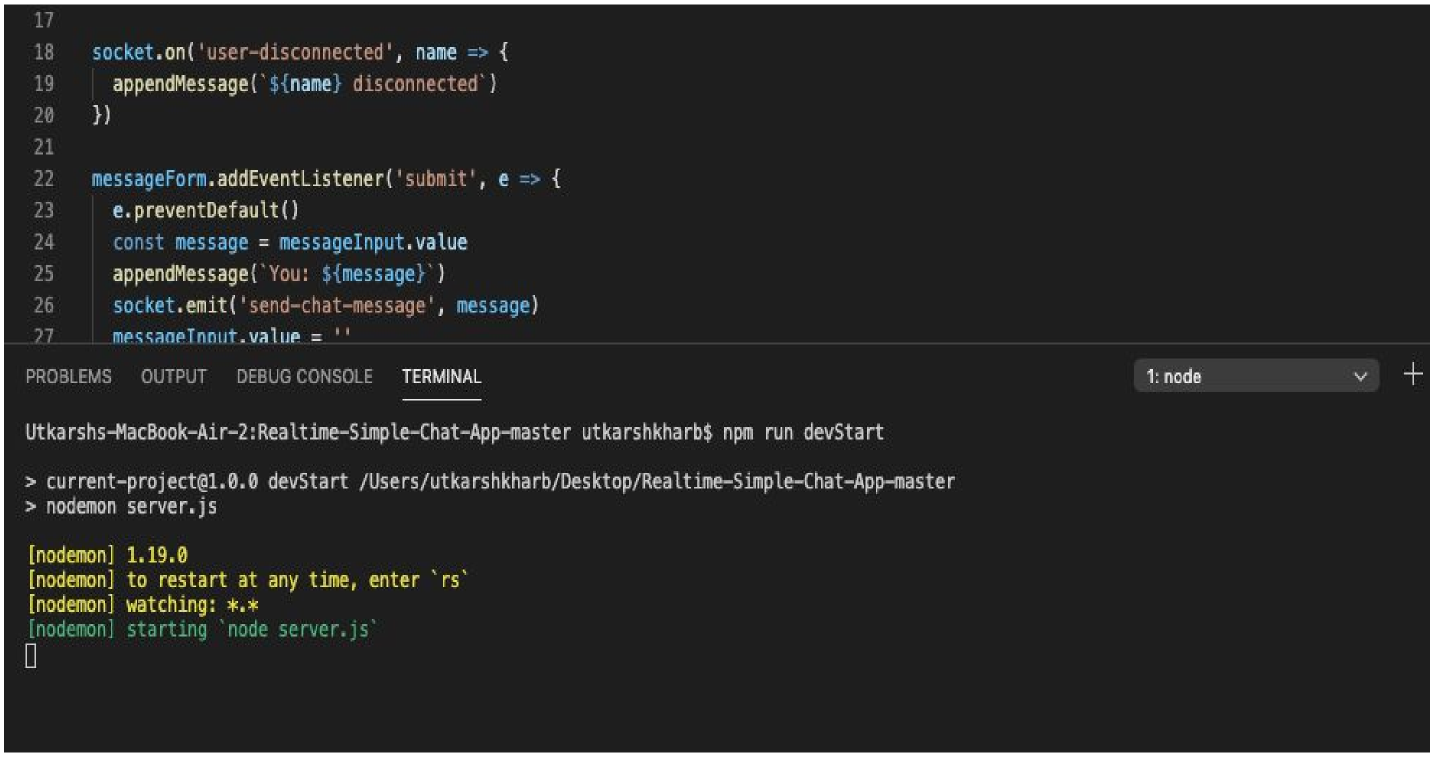
- Operating system like Windows or MacOS or Linux
- Windows 10 is the platform used here to develop 2D & 3D graphics applications.
- A Visual C++ compiler is required for compiling the source code to make the executable file which can then be directly executed.
- Node.js using Express, Mongoose and Socket.io

Hardware requirements

- The hardware requirements are very minimal and the software can be made to run on most of the machines. Processor: Above x86 Processor speed: 500 MHz and above RAM: 64 MB or above storage space 4GB and more Monitor Resolution: A color monitor with a minimum resolution of 640*480 2.3 Platform The package is implemented using Microsoft visual C++ under the windows programming environment.

OUTPUT SNAPSHOTS

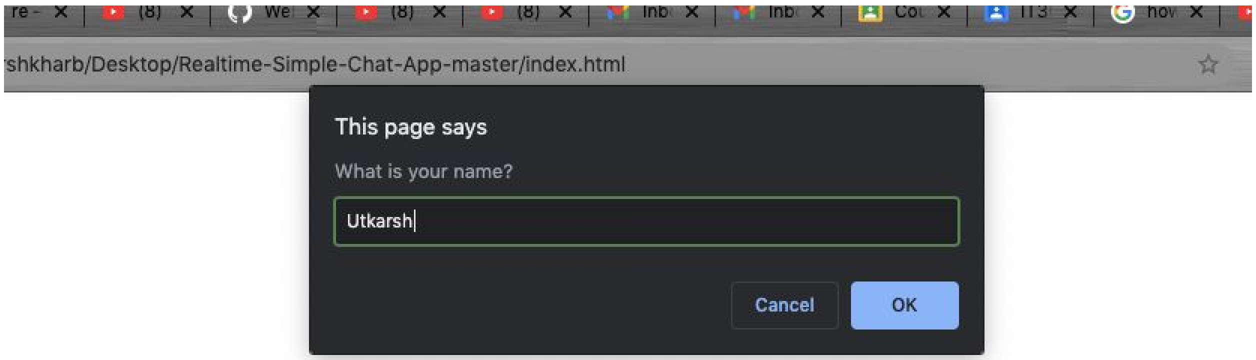
1. **Npm run devStart is used to run the script**



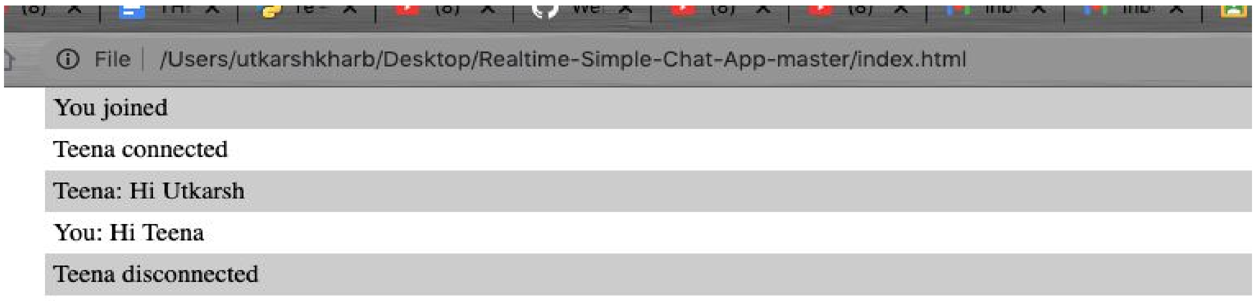
```
socket.on('user-disconnected', name => {
  appendMessage(`${name} disconnected`)
})

messageForm.addEventListener('submit', e => {
  e.preventDefault()
  const message = messageInput.value
  appendMessage(`You: ${message}`)
  socket.emit('send-chat-message', message)
  messageInput.value = ''
```

```
Utkarshs-MacBook-Air-2:Realtime-Simple-Chat-App-master utkarshkharb$ npm run devStart

> current-project@1.0.0 devStart /Users/utkarshkharb/Desktop/Realtime-Simple-Chat-App-master
> nodemon server.js

[nodemon] 1.19.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
```

2. **Index.html file opened in 2 different windows behaves as 2 users as in a chat application.**



If one of the users disconnects, user disconnected is displayed in the chatbox.


File
/Users/utkarshkharb/Desktop/Realtime-Simple-Chat-App-master/index.html

```
You joined
Teena connected
Teena: Hi Utkarsh
You: Hi Teena
Teena disconnected
```

CONCLUSION AND FUTURE WORK

Thus socket.io library of javascript(node js) can be used to make real time chat applications in which users connected to a single server can directly exchange data with each other easily.

For future work, we can take use of cloud application platforms like Heroku to host and deploy the app. The front-end interface of the app can be made more interactive and appealing.

Data can also be encrypted using encryption algorithms like AES and DES to maintain user data privacy.