My project is a chat Application in which multiple users can connect and chat in real time.

Messaging apps are increasing in popularity. Past few years have brought apps like Whatsapp,Telegram etc. Such applications are surrounding us. Messages are sent in an instant and you get real-time feedback. I tried to implement the same functionality and this was the motivation of the project.

It may sound rather complicated without knowing how it works, but in reality, socket.io makes things super convenient.

I have separated things into client side and server side. For the client side, I have-

- **index.html** holding the markup

- **style.css** responsible for the styles

- **client.js** containing the logic for sending messages to server, appending messages in message area,

receiving messages and other functionalities.

and on server side , I have **server.js** containing codes to run server.

This project has 2 dependencies :

- **socket.io** :- socket.io is Javascript library that enables real time, bidirectional communication between browser and server.

- **express.js** :- express.js is web framework for node.js.

Let’s start with index.html. It contains a section tag which is the chat section. Inside the section tag, we have div tag whose class name is brand. It contains brand image and heading. After this there is another div which is the message area. It contains incoming and outgoing messages implemented using DOM. Below this div, there is a textarea tag in which user will write the message and enter key sends the message.index.html file also contains the code for loading socket.io library in client side.

All the styling is done in style.css file in which Roboto font is imported using google fonts.

Coming to client.js file. Firstly, it contains a prompt which asks user name from the user. A do while loop is used to implement this so that until user doesn’t write user name it asks repetitively. The textarea is imported and stored in variable textarea using document.querySelector. As soon as the user hits enter sendMessage function is called. sendMessage function creates a object msg which contains the message and user name. This msg object is first appended in message area using function appendMessage. This appendMessage function creates a div, a markup , user name and msg is added into it and inserted in message area. Now, the sendMessage function sends the message using socket.emit.

On the server side server.js is present. A basic HTTP server is created, using the built-in http module.

Express and socket.io is also instantiated . The http server is started using http.listen on a port.

To serve static files such as images, css files and client.js file express.static built in middleware function is used.

Then real time communication is implemented . In the io variable socket.io library is imported and binded with http server. The io.on event handler establishes the web socket connection between browser and server. Listening a event from client is done using socket.on. After receiving message form client the task was to broadcast the message to all user except one who sent the message which is done is using socket.broadcast.emit.

For listening events on client side code is wriiten in client.js. It listens events using socket.on function which has a callback function which appends the incoming message using appendMessage function.

Let’s talk about **socket.io** – socket.io is not exactly a websocket implementation. It creates a connection between clients and the server. It is a websocket connection , which is bidirectional and connection between server and client is kept alive until it is terminated by either party unlike http connection as soon as client gets the response connection is terminated.

<https://medium.com/weekly-webtips/how-to-make-a-real-time-chat-app-with-socket-io-cf0cae4bc8f2>

<https://www.freecodecamp.org/news/building-a-chat-application-with-mean-stack-637254d1136d/>