Purpose and Evolution of the idea

The main purpose is to create a chat application which provides a common platform for users to interact and discuss. Ajax may be a bit horrific for overall performance, (in the case of chatting) as it’s based totally on a request and reaction. We additionally needed to hold an c programming language in the patron which asked the server for brand spanking new messages and this system become repeated again and again.

Comet is pretty lots the identical. even as the usage of Ajax, we requested the server whenever for messages, with Comet we needed to open one connection and wait. This connection remained open within the server for some time till the server obtained messages for us, then the server responded with the messages and closed the opened connection. The client, when receiving these messages opened a brand new connection and waited once more. One gain of Comet is the reduction of the request quantities, but still, we had to open and close many of them for every consultation (in case the consumer is energetic).

And ultimately, a Flash component - in this example we placed a small flash aspect in our web page to alternate messages with the JavaScript client, the flash ought to open a socket to the server and this gave us push exchange.

### Node.js and Socket.IO

With node.js and socket.io we can enjoy better performance, a bidirectional push verbal exchange among a server and a JavaScript consumer and even the support for all the browsers. Socket.io can work with several transports in order to guide even old browsers.

NodeJS presents the potential to write down back- end code in JavaScript that is the perfect technology for building actual time applications.

With node.js and socket.io we can enjoy better performance, a bidirectional push communication between a server and a JavaScript client and even the support for all the browsers. Socket.io can work with several transports in order to support even old browsers.

NodeJS provides the ability to write back-end code in JavaScript which is the perfect technology for building real time applications.

Technology, Tools used

1. MEAN Stack (Express, Angular, Node)
2. Sockets to enable one-on-one communication in real time- We are able to install a socket on each end and allow a customer to have interaction with other customers through the server. The socket at the server facet friends itself with a few hardware port at the server side. Any client that has a socket associated with the identical port can communicate with the server socket.

Future Prospect of the project:

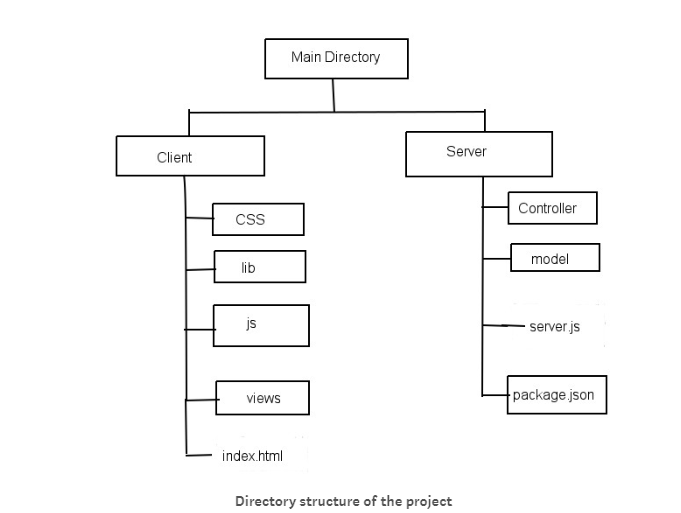
We are also planning to deploy the use of HTML5 APIs for real-time communications such as canvas/video, sockets, getUserMedia, and WebRTC. Along with it we wish to dive into Node.js and learn all about how to use it from the ground up in the command line to communicating with HTML5 in real-time through asynchronous code in Node.js. We also wish to mix up our chat application with Google Maps and the Geolocation API to create a cool geographic app.

Dependencies-

socket.io — It is a javascript library for actual-time web programs. It allows real-time, bi-directional communication between web clients and servers. The consumer-aspect library runs in the browser and the server-facet library for node.js. each components have similar API & like node.js, it is also event-driven. Socket.IO affords the ability to enforce real-time analytics, binary streaming, immediately messaging, and document collaboration.

express — is a Node.js web application framework. It provides the set of features to expand the net and cellular programs. it is easy to respond to HTTP request the use of exceptional middlewares and additionally render HTML pages.

Directory structure and the path to be followed ahead-



* In NodeJS command prompt/terminal we are able to navigate to our software directory i.e. chat-utility.
* Create a package.json file that is the take place report that describes our venture.
* Install express, HTTP, socket.io and path npm modules in our working directory i.e. chat-utility.
* Create an HTML file index.html that is used to serve the chat window within the browser.
* Create a CSS document i.e. style.css to design the chat window.
* Create an index.js file so one can setup our utility on the server.
* Then we will create file for client facet script i.e. chat.js.
* Then go to our web browser after which we will open link in two one of a kind windows to test our chat utility.