**Chat application in Node.js using Express and Socket.io**

### **What is chat application and why?**

### Online chat may refer to any kind of communication over the Internet that offers a real-time transmission of text messages from sender to receiver.

**How does a chat application work?**

Client rapidly breaks up your message into packets, which are delivered directly to the recipient's computer or device. As you chat with your contact, the window appears identically to both parties, and messages appear within a split second of being sent.

### **Node.js**

[Node.js](https://en.wikipedia.org/wiki/Node.js) is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. The most important advantage of using Node is that we can use JavaScript as both a front-end and back-end language.

As we know, JavaScript was used primarily for client-side scripting, in which scripts were embedded in a webpage’s HTML and run client-side by a JavaScript engine in the user’s web browser.

Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting — running scripts server-side to produce dynamic web page content before the page is sent to the user’s web browser.

To install node:

<https://nodejs.org/en/download/>

#### **NPM Modules**

Nodejs allows the modules of libraries to be included in the application. These modules can be user-defined or third party modules.

The third party modules can be installed using the following command:

npm install module\_name

and the installed modules can be used using the **require()** function:

var module = require(‘module\_name’)

In Node apps we will be using a package.json file to maintain the module versions. This file can be created by this command:

npm init

**Description**

The app must allow multiple users to chat together. The messages must be updated without refreshing the page. For simplicity we will be avoiding the authentication part.We can start by creating a new project directory and moving into it. Then we can initiate our project by the following command:

npm init

This will prompt us to enter details about our project.After this a **package.json** file will be created:

{

"name": "chat",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "me",

"license": "ISC",

}

Our app directory is now set.

The first thing we need to create is a server. In order to create that, we will be making use of a framework named **Express.**

#### **Express.js**

Express.js, or simply Express, is a web application framework for Node.js. Express [provides a robust set of features for web and mobile applications](https://expressjs.com/). Express provides a thin layer of fundamental web application features, without obscuring Node.js features.

We will install Express.js using the following command:

npm install -s express

Inside the package.json file a new line will be added:

"dependencies": {

"express": "^4.8.8"

},

**Nodemon**

Nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. nodemon does not require any additional changes to your code or method of development

We will install nodemon using the following command:

npm install -g nodemon

-g — global, so that it is accessible in all projects.

Inside the package.json file a new line will be added:

"devDependencies": {

"nodemon": "^1.19.1"

}

Next we will create a **index.js** file.

In this file we need to require Express and create a reference to a variable from an instance of Express. Static contents like HTML, CSS or JavaScript can be served using express.js:

var express = require('express');

var app=express()

and we can start listening to a port using the code:

var server=app.listen(4000,function(){

console.log('listening to requests on port 4000')

});

Now we need to create an HTML file index.html that displays our UI.

So have create a static folder name public in that files index.html,styles.css and chat.js located

app.use(express.static('public'));

We can run the server.js using the command:

Nodemon index.js

If you go to localhost:4000 we can see the index file.

The only issue now is that there is no way for the client to know if the server is updated. So each time we post a message we need to refresh the page to see the new messages.

To solve this we can add a push notification system that will send messages from server to client. In Node.js we use **socket.io.**

#### **Socket.io**

Socket.IO is a JavaScript library for realtime web applications. [It enables realtime, bi-directional communication between web clients and server](https://www.tutorialspoint.com/socket.io/socket.io_overview.htm). 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.

To install socket.io:

npm install socket.io --save

Code to **index.js** and we can create a connection

var express = require('express');

var socket = require('socket.io');

var app=express()

var server=app.listen(4000,function(){

console.log('listening to requests on port 4000')

});

app.use(express.static('public'));

var io = socket(server);

In the **index.html**add the following tag which is taken from socket.io site:

<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.js">

</script>

We can run the server.js using the command:

Nodemon index.js

If you go to **localhost:4000** we can see the index file.