

WEB BASED CHAT APPLICATION

A Summer Internship Project Report

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

SHAURYA LALWALIA, 04020902718

**Guided by
Mr. Manjeet Pangtey**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

G.B. PANT GOVT. ENGINEERING COLLEGE, NEW DELHI.

(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)

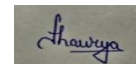
DECEMBER, 2021

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the Summer Internship/Training entitled “**Web Based Chat Application**” in partial fulfilment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering / Information Technology affiliated to **Guru Gobind Singh Indraprastha University, New Delhi** and submitted to the Department of Computer Science and Engineering G. B. Pant Govt. Engineering College , is an authentic record of my own work. The matter represented in this report has not been submitted by me for award of any other degree of this or any other institute/university.

Date – 31th Dec, 2021

Name – Shaurya Lalwalia
(04020902718)



This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date -

Signature of Supervisor
(Mr. Manjeet Pangtey)

TABLE OF CONTENTS

| <u>CHAPTER NO.</u> | <u>TITLE</u> | <u>PAGE NO.</u> |
|---------------------------|---|------------------------|
| | 1) Title Page | 1 |
| | 2) Candidate Declaration | 2 |
| | 3) List of Figures | 5 |
| | 4) Acknowledgement | 6 |
| | 5) Abstract | 7 |
| Chapter 1. | INTRODUCTION | 8 - 9 |
| | 1.1) Problem Statement | 9 |
| | 1.2) Objectives | 9 |
| Chapter 2. | TECHNOLOGIES USED | 10 - 12 |
| | 2.1) What is MongoDB? | 10 |
| | 2.2) What is Express.js | 10 |
| | 2.3) What is React.js | 11 |
| | 2.4) What is Node.js | 12 |
| Chapter 3. | SYSTEM DESIGN & IMPLEMENTATION | 13 - 18 |
| | 3.1) GUI Interface | 13 |
| | 3.2) Implementation | 14 |
| | 3.2.1) Chat App or Client Side | 15 |
| | 3.2.2) Chat Server Engine | 16 |
| | 3.2.3) Web Socket | 17 |
| Chapter 4. | RESULTS | 19 - 21 |
| | 4.1) Screenshots | 19 |
| | 4.1.1) Setting up the Connection | 19 |
| | 4.1.2) Entering Interface | 20 |
| | 4.1.3) Chatting Demo | 21 |

| | | |
|------------|-------------------|-----------|
| Chapter 5. | CONCLUSION | 22 |
| Chapter 6. | REFERENCES | 23 |

LIST OF FIGURES

| | |
|--|----|
| • Figure 1 - MERN Stack..... | 8 |
| • Figure 2 - GUI Interface..... | 13 |
| • Figure 3 - GUI Code & Styling..... | 14 |
| • Figure 4 - Client Side Structure Tree..... | 15 |
| • Figure 5 - Package.json file of Client Side..... | 16 |
| • Figure 6 - Dependencies of Server..... | 17 |
| • Figure 7 - Socket Code..... | 18 |
| • Figure 8 - Activating the Port..... | 19 |
| • Figure 9 - localhost..... | 20 |
| • Figure 10 - Input your Name..... | 20 |
| • Figure 11 - Chatting Demo..... | 21 |

ACKNOWLEDGEMENT

I would like to express our gratitude towards our mentor Mr. Manjeet Pangtey for exposing me to this topic and moreover for being the guiding light all this while. Furthermore I would like to thank other teachers for their valuable suggestions and to our principal for presenting me with this golden opportunity of making this project. Finally, I would like to thank my friends and family for being the constant support system and encouraging force all this while.

ABSTRACT

Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance was quite recent. Our project is an example of a chat server. It is made up of two applications - the client application, which runs on the user's web browser and server application, runs on any hosting servers on the network. To start chatting client should get connected to server where they can do private and group chat.

CHAPTER-1

INTRODUCTION

Today Developers around the world are making efforts to enhance user experience of using application as well as to enhance the developer's workflow of designing applications to deliver projects and rollout change requests under strict timeline. Stacks can be used to build web applications in the shortest span of time. The stacks used in web development are basically the response of software engineers to current demands. They have essentially adopted pre-existing frameworks (including JavaScript) to make their lives easier.

While there are many, MEAN and MERN are just two of the popular stacks that have evolved out of JavaScript. Both stacks are made up of open source components and offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. The common theme between the two is JavaScript and this is also the key benefit of using either stack. One can basically avoid any syntax errors or any confusion by just coding in one programming language, JavaScript. Another advantage of building web projects with MERN is the fact that one can benefit from its enhanced flexibility.

In order to understand MERN stack, we need to understand the four components that make up the MERN stack(fig.1), namely – MongoDB, Express.js, React and Node.js.

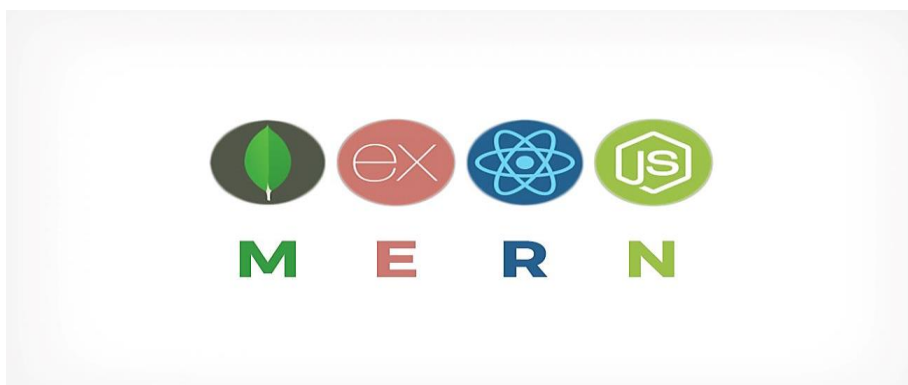


Figure 1 : MERN Stack

1.1) Problem Statement

- This project is to create a chat application with a server and users to enable the users to chat with each other's.
- To develop an instant messaging solution to enable users to seamlessly communicate with each other.
- The project should be very easy to use enabling even a novice person to use it.
- This project can play an important role in organizational field where employees can connect through LAN.

1.2) Objectives

- **GUI:** Easy to use GUI (Graphical User Interface), hence any user with minimal knowledge of operating a system can use the software.
- **Platform independence:** The messenger operates on any system irrelevant of the underlying operating system.
- **Unlimited clients:** “N” number of users can be connected without any performance degradation of the server.

Objective is to develop an instant messaging solution to enable users to seamlessly communicate with each other. Furthermore, it needs to be user-friendly, i.e., the project should be very easy to use enabling even a novice person to use it.

CHAPTER-2

TECHNOLOGIES USED

2.1) What is MongoDB? [1]

- MongoDB is a cross-platform document-oriented NoSQL database used for high volume data storage that provides high performance, high availability and easy scalability.
- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in the application code, making data easy to work with.
- The data model available within MongoDB allows users to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
- MongoDB works on concept of collections and documents. Each database contains collections which in turn contains documents. Each document can have varying number of fields. The size and content of each document can also be different from each other.

2.2) What is Express.js ? [2]

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the Node.js foundation.
- Express provides us the tools that are required to build our app, be it single-page, multi-page or hybrid web applications. It is flexible as there are numerous modules available on **npm(Node Package Manager)**, which can be directly plugged into Express.

- Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.
- Pug (earlier known as Jade) is a terse language for writing HTML templates. It produces HTML, supports dynamic code and code reusability (DRY). It is one of the most popular template languages used with Express.
- Express can be thought of as a layer built on the top of the Node.js that helps manage a server and routes. It allows users to setup middleware to respond to HTTP Requests and defines a routing table which is used to perform different actions based on HTTP method and URL.
- Express allows to dynamically render HTML Pages based on passing arguments to templates.
- Express is asynchronous and single threaded and performs I/O operations quickly.

2.3) What is React ? [3]

- ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.
- A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.
- Instead of using regular JavaScript, React codes are written in something called JSX (JavaScript Syntax Extension). JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM. JSX is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript.

2.3.1) Why use React? [4]

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

2.4) What is Node.js? [5]

- Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single- page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.
- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009.
- Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.
- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

2.4.1) Features of Node.js

1. Extremely fast: Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. I/O is Asynchronous and Event Driven: All APIs of Node.js library are asynchronous i.e. non-blocking. So, a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.
3. Single threaded: Node.js follows a single threaded model with event looping.
4. Highly Scalable: Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
5. Open source: Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js application.

CHAPTER-3

SYSTEM DESIGN & IMPLEMENTATION

This is a very basic web based chat application and it is in its primary state. I have provided with a very easy to use GUI Interface.

3.1) GUI Interface

This is the GUI interface made using React.js etc

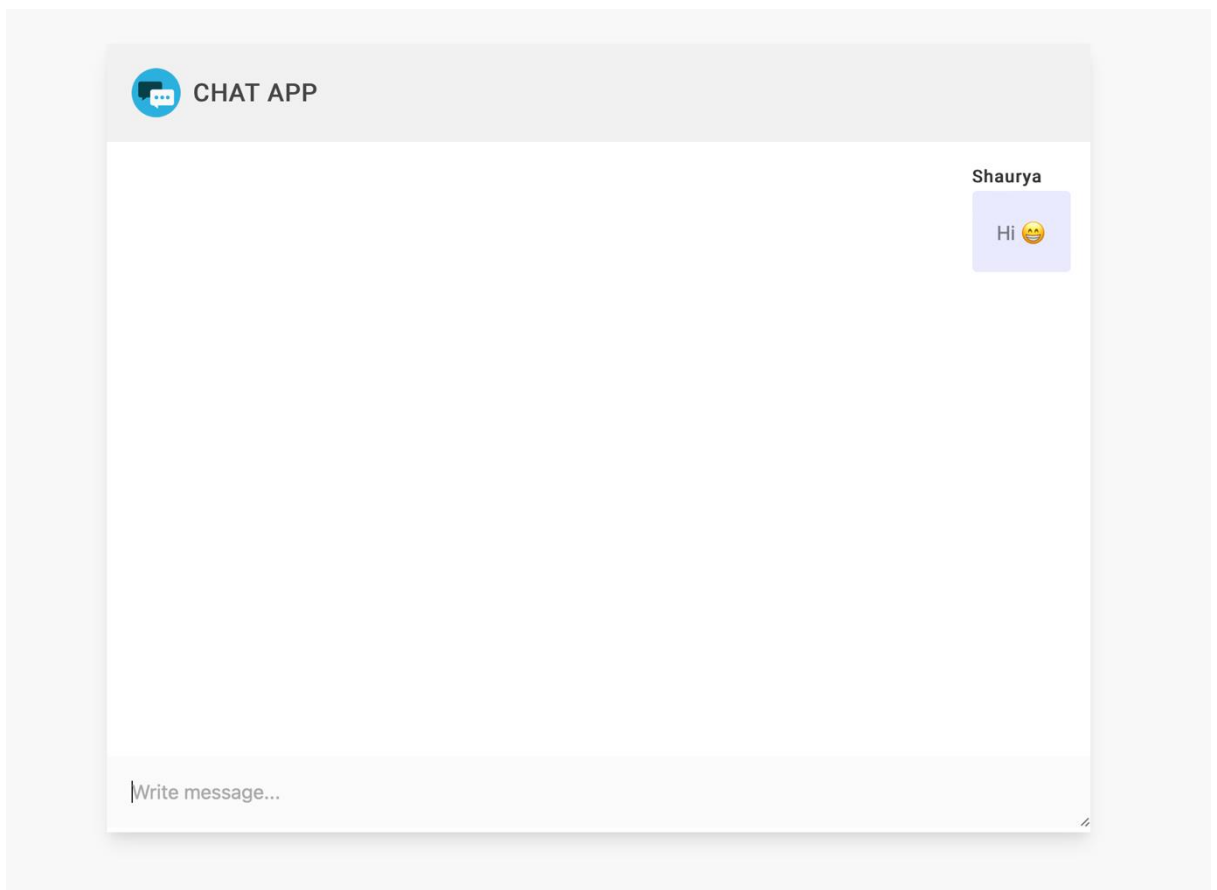


Figure 2 : GUI Interface

CODE :-

```
<section class="chat_section">

  <div class="logo">
    
    <h1>Chat App</h1>
  </div>
  <div class="msg_area"></div>

  <div>
    <textarea id="textarea" cols="30" rows="1" placeholder="Write message..."></textarea>
  </div>
</section>
```

```
section.chat_section {
  width: 800px;
  max-width: 90%;
  background: #fff;
  box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);
}

.logo {
  padding: 20px;
  background: #f1f1f1;
  display: flex;
  align-items: center;
```

Figure 3 : GUI Code & Styling

3.2) Implementation

A chat consists of two major parts:

- **Chat App** or **client part**, which is a Web chat application. Build on React
- **Chat Server Engine** or **server part**, which is a pool of external servers responsible for the chat operation. This is the place where all the chat magic happens. Both parts contain various components that communicate to each other and bring the chat into action

3.2.1) Chat App or Client Side

Chat App is the other major part of the chat architecture, the one that users directly interact with. It's split into two separate root components:

- Chat Client Engine handles all the communication with the Chat Server Engine via its internal components: Chat REST API Client Library and Chat WebSocket Client Library.
- Chat UI displays data to users: Chat Contact List UI, Chat Dialog UI

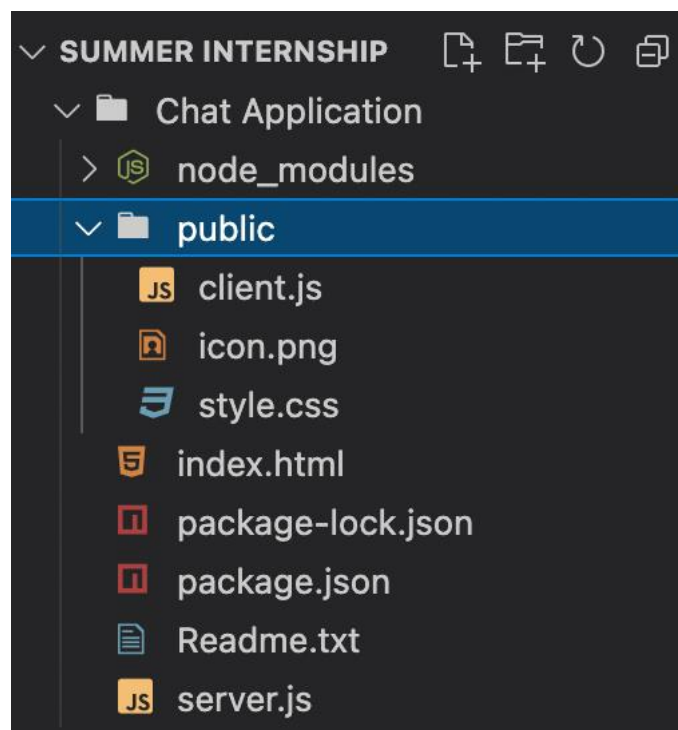


Figure 4 : Client-Side Structure Tree

Component:

client.js is the starting point of our React app.

A **package.json** file :

- lists the packages your project depends on

- specifies versions of a package that your project can use.
- makes your build reproducible, and therefore easier to share with other developers.

A **package.json** file may look similar to this:

CODE :-

```
"name": "chat-application",
"version": "1.0.0",
"description": "This is a real-time chat application",
"main": "index.js",
  > Debug
"scripts": {
  |   "dev": "nodemon server"
  },
```

Figure 5 : Package.json file of Client-side

3.2.2) Chat Server Engine

This is a core of the chat architecture that handles message delivery and dispatch. In our version of chat architecture, it includes the following components:

- Chat REST API handles the tasks that are not connected directly to message dispatch and delivery, such as user authentication, changing of user settings, friends invitation, downloading sticker packs, etc. The Chat App (the chat client part) communicates with the Chat REST API via the Chat REST API Client Library.
- Chat WebSocket Server is responsible for transmitting messages between users. The Chat App communicates with the Chat WebSocket Server via the Chat WebSocket Client Library. This connection is open two ways; that means users don't have to make requests to the server if there are any messages for them, they just get them right away.

CODE :-

```
"name": "chat-application",
"version": "1.0.0",
"lockfileVersion": 2,
"requires": true,
"packages": {}
  "": {
    "name": "chat-application",
    "version": "1.0.0",
    "license": "ISC",
    "dependencies": {
      "express": "^4.17.1",
      "socket.io": "^4.1.3"
    },
    "devDependencies": {
      "nodemon": "^2.0.12"
    }
  }
```

Figure 6 : Dependencies for Server

3.2.3) Web Socket

WebSocket is a popular communication protocol (TCP) that enables seamless full-duplex communication (two-way communication) between client and server. In other words, it allows websites send and receive data without delay. WebSockets do not use the http:// or https:// scheme (because they do not follow the HTTP protocol). Instead, WebSocket URIs use a new scheme ws: or wss: (recommended) for a secure WebSocket. Web developers use WebSocket to build chat applications, multiplayer games, SDKs, or user interfaces exposing server-side services in real-time and so much more.[6]

CODE :-

```
//socket code

// import the socket.io and initialise with http server
const io = require('socket.io')(http)

//as soon as browser connects, socket.io detects the connection
io.on('connection', function(socket){
  console.log("connected")

  //listen the emitted message from client.js, msg = message from client
  socket.on('message', function(msg){
    //we have to send this msg to all the browsers/clients
    socket.broadcast.emit('message', msg) //sends the msg to everyone except the one who wrote it
  })
})
```

Figure 7 : Socket Code

CHAPTER-4

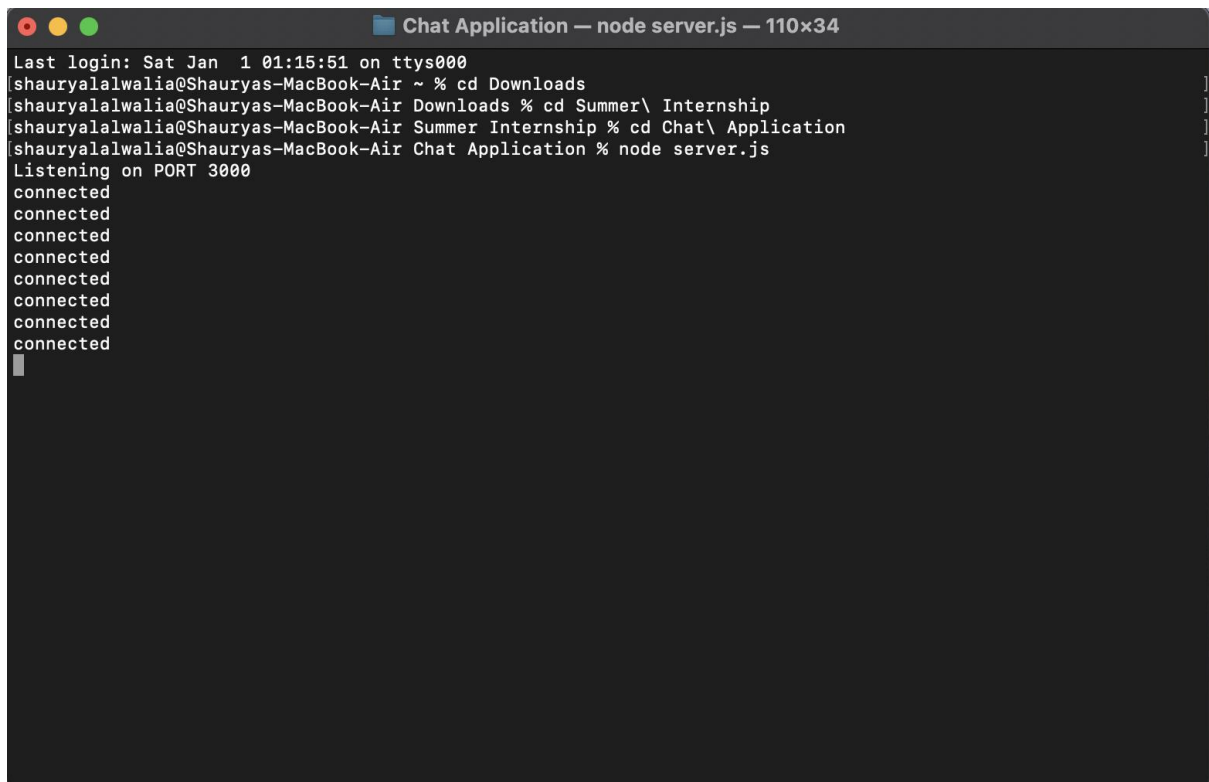
RESULTS

4.1) Screenshots

Below are the screenshots of our web based chat application that demonstrate its functioning :-

4.1.1) Setting up the Connection

This images displays how the connection is build and Port 3000 is active.

A terminal window titled "Chat Application — node server.js — 110x34" is shown. The terminal displays the following text:

```
Last login: Sat Jan 1 01:15:51 on ttys000
shauryalalwalia@Shauryas-MacBook-Air ~ % cd Downloads
shauryalalwalia@Shauryas-MacBook-Air Downloads % cd Summer\ Internship
shauryalalwalia@Shauryas-MacBook-Air Summer Internship % cd Chat\ Application
shauryalalwalia@Shauryas-MacBook-Air Chat Application % node server.js
Listening on PORT 3000
connected
connected
connected
connected
connected
connected
connected
connected
connected
connected
```

Figure 8 : Activating the Port

4.1.2) Entering Interface

This image display that when to visit the localhost at Port 3000, you have to give input your name.



Figure 9 : localhost

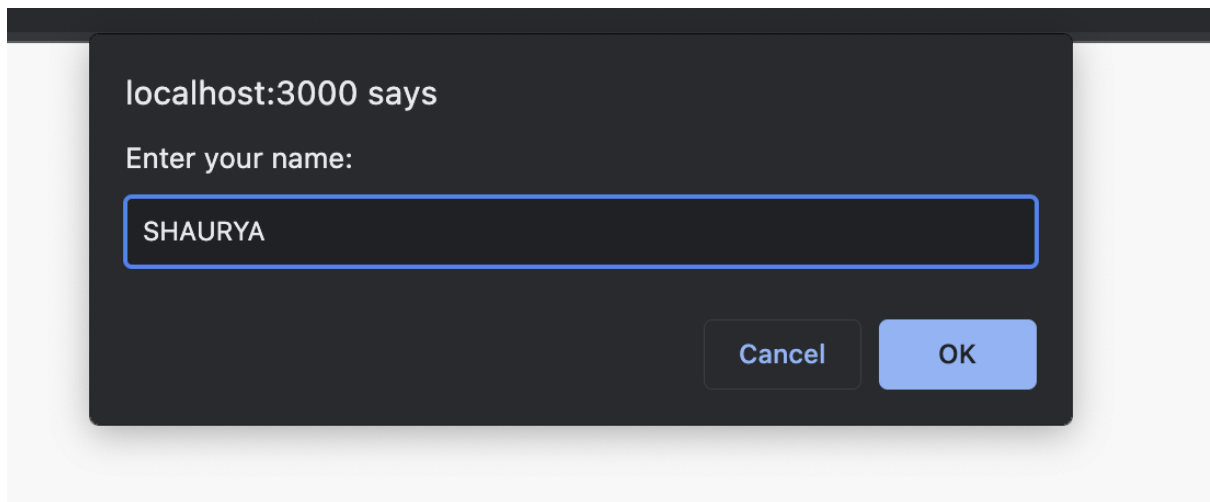


Figure 10 : Input your Name

4.1.3) Chatting Demo

Here is a depiction how your send and received messages will appear on the respective screens.

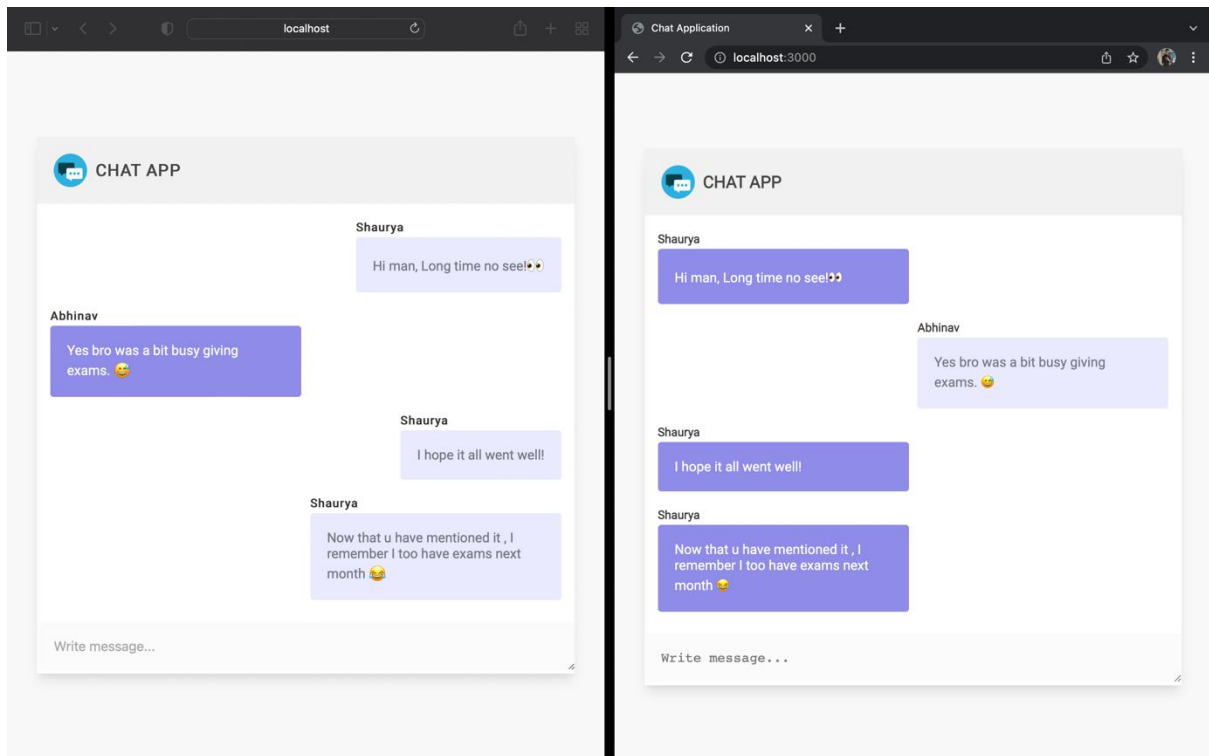


Figure 11 : Chatting Sample

CHAPTER-5

CONCLUSION

There is always a room for improvements in any apps and this being in its primary phase needs a loads of it. But I will keep working on it and improving it with technological advancements in coming times. Right now, we are just dealing with text communication. There are several chat apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. I have built this application keeping in mind that even the very beginner with technology can operate this application with ease. A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a chat service Web app with high quality user interface. In future we may be extended to include features such as file transfer, video message, voice message, audio and video calls etc.

CHAPTER-6

REFERENCES

1. <https://www.guru99.com/what-is-mongodb.html>
2. <https://www.javatpoint.com/expressjs-tutorial>
3. <https://www.javatpoint.com/reactjs-tutorial>
4. https://www.tutorialspoint.com/reactjs/reactjs_overview.htm
5. <https://www.javatpoint.com/nodejs-tutorial>
6. <https://www.cometchat.com/tutorials/how-to-build-a-chat-app-with-websockets-and-node-js>