PROJECT REPORT

CHAT APP

SUBMITTED BY,

RAHANA G KRISHNAN

ADIT/TVM/19/011

ADIT (2019-2021)

NSTI (W), TRIVANDRUM

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.

CONTENTS

1. INTRODUCTION

Today Developers around the world are making efforts to enhance user experience of using application as well as 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.

1.1Objective/Project Overview

To develop an instant messaging solution to enable users to seamlessly communication.

1.2 Project 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.

1.3 Scope of Work

The purpose of the chat application is to allow users be able to the chat with each other, like a normal chat application.

2. SOFTWARE DEVELOPMENT ENVIRONMENT

We use the Node.js platform to build a **real time chat application** that sends and shows messages to a recipient instantly without any page refresh. We use the JavaScript framework Express.js and the libraries Mongoose and Socket.io to achieve this.

3. SYSTEM REQUIREMENTS

3.1 SOFTWARE SPECIFICATION

• Operating System

• Browser-Microsoft Edge/Chrome/Mozilla

• Text Editor-Visual Studio Code /Notepad

3.2 HARDWARE SPECIFICATION

- RAM: 4 GB or above
- Processor: 1 GHz or more
- Hard Drive: 1TB or above
- Network Connectivity: Wi-Fi

4. APPENDICES

4.1 SOURCE CODE

1. Index.js

```
const express = require ('express')
const http = require ('http')
const socketio = require ('socket.io')
const path = require ('path')
const ejs = require ('ejs')

const app = express ()
const server = http.createServer (app)
const io = socketio (server)

const messages = []

app.use (express.static (path.join (__dirname, 'public')))
app.set ('views', path.join (__dirname, 'public'))
app.engine ('html', ejs.renderFile)
app.set ('view engine', 'html')

app.get ('/', (request, response) => {
   response.render ('index.html')
})

io.on ('connection', socket => {
   console.log (`User login: ${socket.id}`)

   socket.emit ('previousMessages', messages)
```

```
    socket.on('sendMessage', (message) => {
        messages.push (message)
        socket.broadcast.emit('recivedMessage', message)
    })
})

const port = process.env.PORT || 3000;
server.listen (port, () => {
    console.log (`> Server listening on port ${port}`)
})
```

2. index.html

```
<! DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Chat App</title>

        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div id="root">
            <header>
                <h1>Personal Chat App</h1>
                <div id="user-form">
                    <label for="username">username</label>
```

```
                <input type="text" id="username" placeholder="Type your username...">
            </div>
        </header>
        <ul id="messages"></ul>
        <form id="chat-form">
            <input type="text" autocomplete="off" id="message-text" placeholder="Type your message...">
            <button type="submit">&vartriangleright;</button>
        </form>
    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.0/socket.io.js"></script>
    <script>
        const socket = io()
        const usernameInput = document.querySelector("#username")
        const savedUsername = localStorage.getItem('capivara-username')
        const form = document.querySelector('#chat-form')
        const messageInput = document.querySelector ('#message-text')
        const messagesContainer = document.querySelector ('#messages')

        if (savedUsername){
            usernameInput.value = savedUsername
        }

        function saveUsername (){
            localStorage.setItem ('chater-username', usernameInput.value)
        }
```

```
form.addEventListener ('submit', (event) => {
    event.preventDefault()
    const currentUsername = usernameInput.value
    const message = messageInput.value

    if (!currentUsername || !message) return

    const messageObject = {author: currentUsername, text: messa
ge}

    socket.emit ('sendMessage', messageObject)
    saveUsername()
    messageInput.value = ''
    appendMessage (messageObject)
})

function appendMessage (message) {
    const messageAuthor = document.createElement ("p")
    messageAuthor.classList.add('author')
    messageAuthor.innerHTML = message.author

    const messageText = document.createElement("p")
    messageText.classList.add ("text")
    messageText.innerHTML = message.text

    Const li = document.createElement ('li')
    li.appendChild(messageAuthor)
    li.appendChild(messageText)

    if (message.author == usernameInput.value) {
        li.classList.add ("sent-message")
    } else {
```

```
            li.classList.add("recived-message")
        }

        messagesContainer.appendChild(li)
    }

    socket.on('recivedMessage', (message) => {
        appendMessage (message)
    })

    socket.on('previousMessages', (messages) => {
        for(message of messages){
            appendMessage(message)
        }
    })
</script>
</body>
</html>
```

3. **style.css**

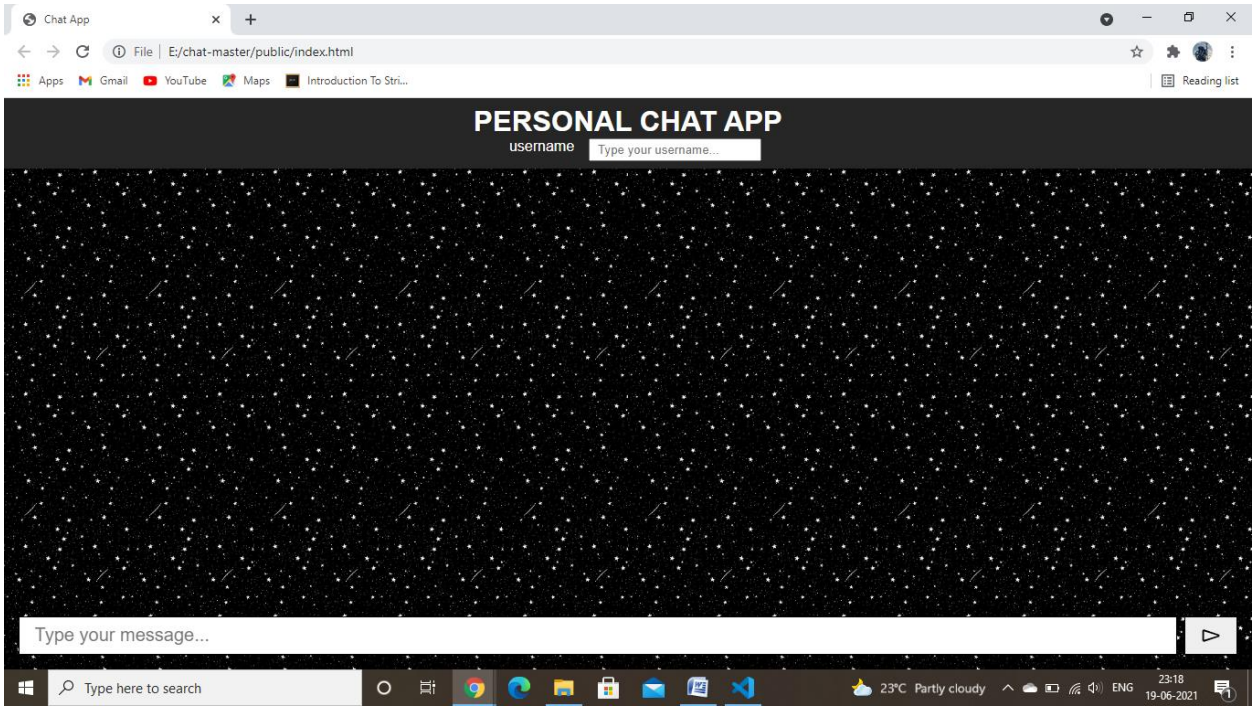
```
html, body, #root {
    width: 100%;
    height: 100%;
    padding: 0;
    margin: 0;
    font-family: 'Open Sans', sans-serif;
    background-image: url ('bg.jpg');
    background-size: 10%;
    background-blend-mode: multiply;
}
```

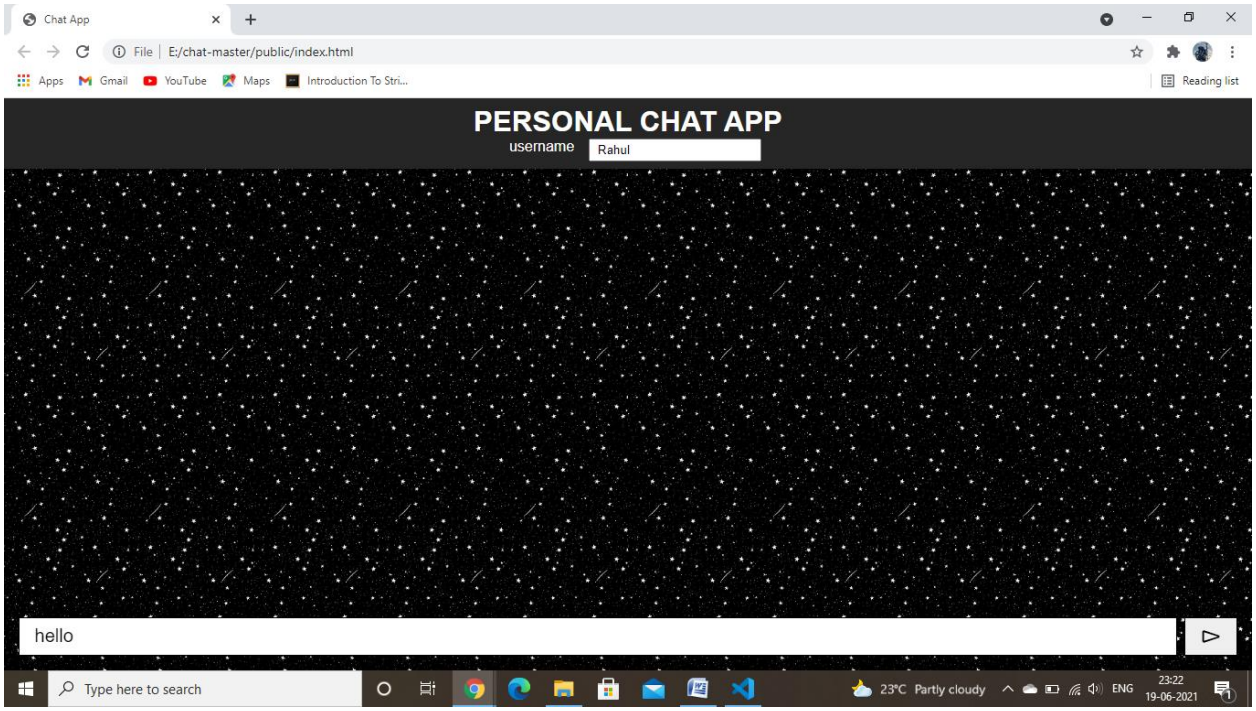
```
input: focus, button: focus{
    outline: none;
}

header {
    padding: 0.5em;
    background-color: #262626;
    color: #ffffff
}

header h1{
    margin: 0;
    text-align: center;
    text-transform: uppercase;
    font-size: 1.953rem;
}

#user-form {
    display: flex;
    Justify-content: center;
}

#user-form label {
    padding: 0 1rem;
}

#user-form input {
    padding: 0.2rem 0.5rem;
}

#messages {
    list-style-type: none;
```

```
    margin: 0;
    padding: 0 0.5rem;
}

#messages li{
    border-radius: 1rem;
    border: 0.1rem solid black;
    box-sizing: border-box;
    padding: 0.5rem;
    margin: 0.5rem 0;
    max-width: 70%;
}

#messages li.recived-message {
    background-color: #ffffff;
}

#messages li.sent-message{
    background-color: rgb (255, 226, 188);
    margin-left: 30%;
}
p.author {
    font-weight: bold;
    margin-top: 0;
}

p.text {
    margin-bottom: 0;
}

#chat-form {
    position: fixed;
```

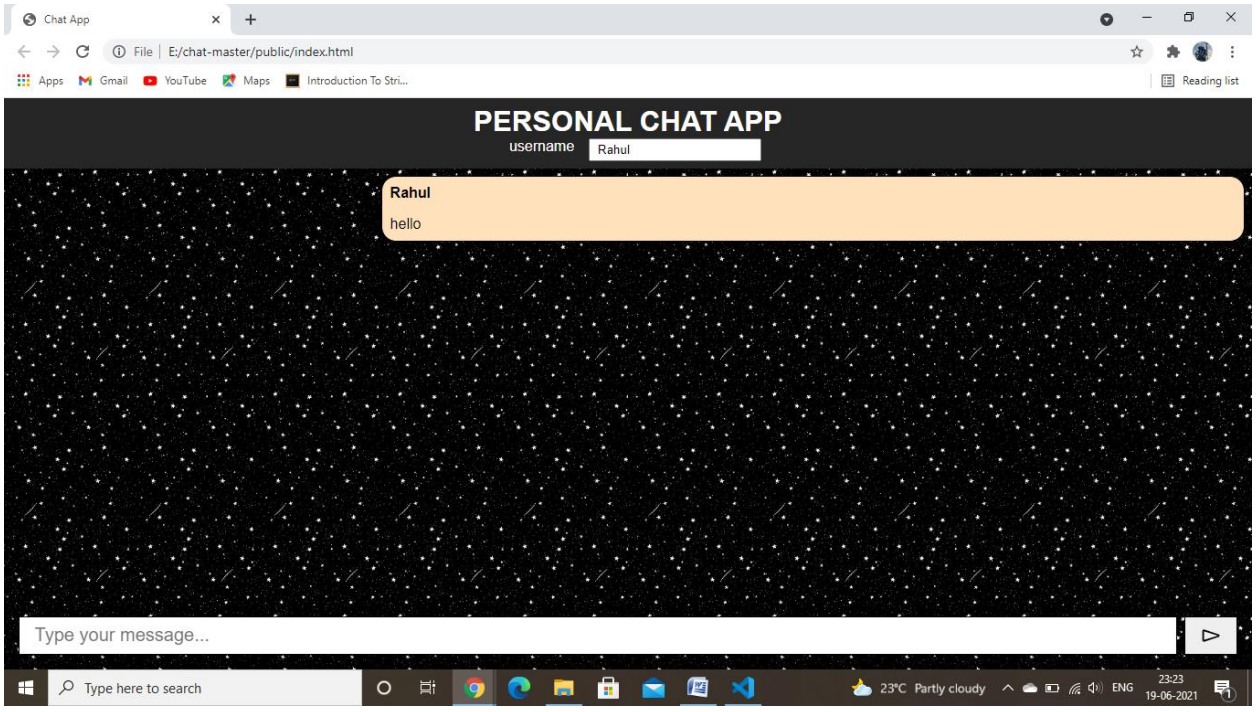
```
    bottom: 0;
    width: 100%;
    display: flex;
    /* background-color: cornflower blue; */
    padding: 1rem;
    box-sizing: border-box;
}

#chat-form input {
    border: 0.1rem solid black;
    width: 100%;
    font-size: 1.25rem;
    padding-left: 1rem;
}

#chat-form button {
    border: 0.1rem solid black;
    margin-left: 0.5rem;
    font-size: 2rem;
    padding: 0 1rem;
    cursor: pointer;
}
```

4.2 OUTPUT


Chat App
File | E:/chat-master/public/index.html
Apps
Gmail
YouTube
Maps
Introduction To Stri...
Reading list
PERSONAL CHAT APP
username
Type your username...
Type your message...
Type here to search
23°C Partly cloudy
ENG
23:18
19-06-2021


Chat App
File | E:/chat-master/public/index.html
Apps
Gmail
YouTube
Maps
Introduction To Stri...
Reading list
PERSONAL CHAT APP
username
Rahul
hello
Type here to search
23°C Partly cloudy
ENG
23:22
19-06-2021

4.3 RESULT


Chat App
File | E:/chat-master/public/index.html
Apps
Gmail
YouTube
Maps
Introduction To Stri...
Reading list
PERSONAL CHAT APP
username
Rahul
Rahul
hello
Type your message...
Type here to search
23°C Partly cloudy
ENG
23:23
19-06-2021

5. CONCLUSION

There is always a room for improvements in any apps. Right now we are just dealing with text communication. There are several android apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. A positive first impression is essential in human relationship as well as in human computer interaction.

6. REFERENCE

- https://www.freecodecamp.org/news/simple-chat-application-in-node-js-using-express-mongoose-and-socket-io-ee62d94f5804/
- https://blog.crowdbotics.com/build-chat-app-with-nodejs-socket-io/