

# **TABLE OF CONTENTS**

## **1. ABSTRACT**

## **2. INTRODUCTION**

2.1. Introduction

2.2. Problem Statement

2.3. Innovative ideas of project

2.4. Aim and objectives

2.4.1. Messaging

2.4.2. File transfer

2.4.3. Deliverables

2.5. Background and motivation

2.6. What is Express.js?

2.7. What is Node.js?

2.8. What is Socket.IO?

## **3. REQUIRIMETS AND DESIGN**

3.1. Hardware Requirement

3.2. Software Requirement

3.3. Libraries

3.4. Group Chat Interface

3.5. Chat server Engine

## **4. SCREENSHOTS**

## **5.CONCLUSION**

## **6. REFERENCES**

# ABSTRACT

Group 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. My project is an example of a group 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 group chat and share multimedia content with other members of the group. So the name which I have given to the project is **Sky-Chat**.

Sky-Chat is a social-networking tool that leverages on technology advancement thereby allowing its user's to communicate with multiple people at the same time and share media , therefore can be used for messaging, share updates and photos, enhance local socializing in pidgin English. This app was made keeping one thing in mind that the UI should be smooth and for creating the groups that are for short span of time because people used to make a lot of groups and are not even active in that group so with Sky-Chat, the group would be available by the time the user logout and with it people can be free from the fear that their data is being tracked or stored because in Sky-Chat we are not storing any messages or media of the user, the thing we are storing is user login information.

# INTRODUCTION

Sky-Chat is created with the mind set to add a level of clean UI (user interface) or a solid app structure function over a secure broadcast network. Sky-Chat is an effort for a more modern approach to internet security on a communication medium. The Implementation of Sky-Chat is based on HTML, CSS, JS, NODEJS and SOCKET IO where HTML, CSS, JS is for the app structure and UI (user interface) tool, NODEJS and SOCKET IO for the server setup. An end to end connection stream was used for data transfer from client to server and back to client. In conclusion the application worked well without a lag and having a 95% acceptance when tested with a potential user.

## PROBLEM STATEMENT

Developing a group-chat web application: The web application should allow users to chat with each other and share photos and other multimedia content.

Starting any application or service has many problems but one of the main problems is which tool, language, stack or framework to build one's service or application on. As building a real time application has to do with slow latency message delivery which in turn means latency, data transfer size over the network must be as low as possible.

## INNOVATIVE IDEAS OF PROJECT

**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.

**GUI:** Easy to use GUI (Graphical User Interface), hence any user with minimal knowledge of operating a system can use the software.

# **AIMS AND OBJECTIVES**

## **MESSAGING :**

One of the primary use of SKY-CHAT is messaging. This feature is pivotal as people can join the group and start chatting.

This is made possible as each of the SKY-CHAT user will have a unique login id and password and once the user log in we provide the user to choose his username to join the chat and by entering the group name and group password.

And once the user is a part of the group than he/she can chat with their friends and share the multimedia content.

And this can be achieved by using socket io for creating the different rooms for the different groups

## **FILE TRANSFER :**

With the sophistication design of the SKY-CHAT , individuals will be able to share files to the group without size constraints ranging from images, videos, to large documents files like zip, dmg, and so on.

## **DELIVERABLES :**

There are 3 major things I hope to achieve with this application, which include.

- I. Privacy Protection
- II. Speed in usage
- III. Easy and friendly UI

# BACKGROUND AND MOTIVATION

The Group Chat Application we are currently using are not that much secure because they are storing the user messages and media although they are claiming that the data is completely encrypted but that's not the point they have the access to the private messages of the users and their media.

So keeping that thing in mind I created SKY-CHAT because with it people can be free from the fear that their data is being tracked or stored because in SKY-CHAT we are not storing any messages or media of the user, the thing we are storing is user login information. We come across this incident every month that we created groups and after sometime no one is active in the group and the group is completely inactive so, in SKY-CHAT the groups are created for short span of time until user logout. This is done to overcome this problem.

## What is Express.js?

- ✓ 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.

## Why use Express?

- ✓ Ultra-fast I/O.
- ✓ Asynchronous and single threaded.
- ✓ MVC like structure.
- ✓ Robust API makes routing easy.

## What is Node.js?

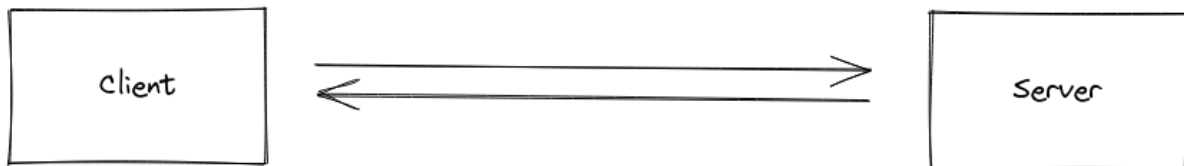
- ✓ Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single- page applications, and other web applications. Node.js is open source, completely free, and used by thousands 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.

## Features of Node.js

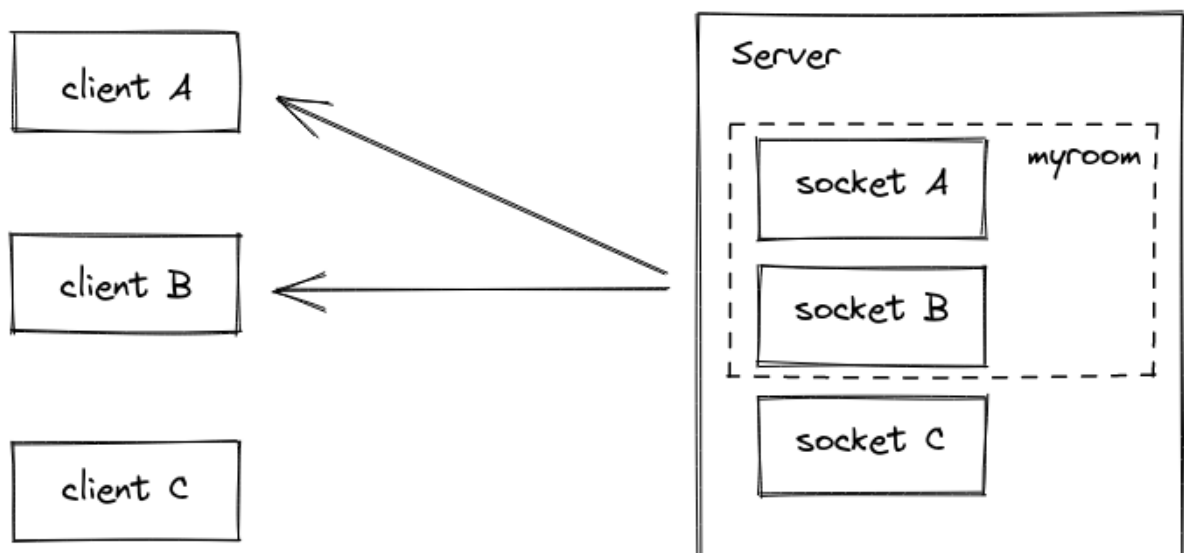
1. **Extremely fast** : Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. **I/O is Asynchronous and Event Driven** : All APIs of Node.js library are asynchronous i.e. non-blocking. So, a Node.js based server never waits for an API to return data. 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. **No buffering** : Node.js cuts down the overall processing time while uploading audios videos and files. Node.js applications never buffer any data. These applications simply output the data in chunks.
6. **Open source** : Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js application.

## What is Socket.IO?

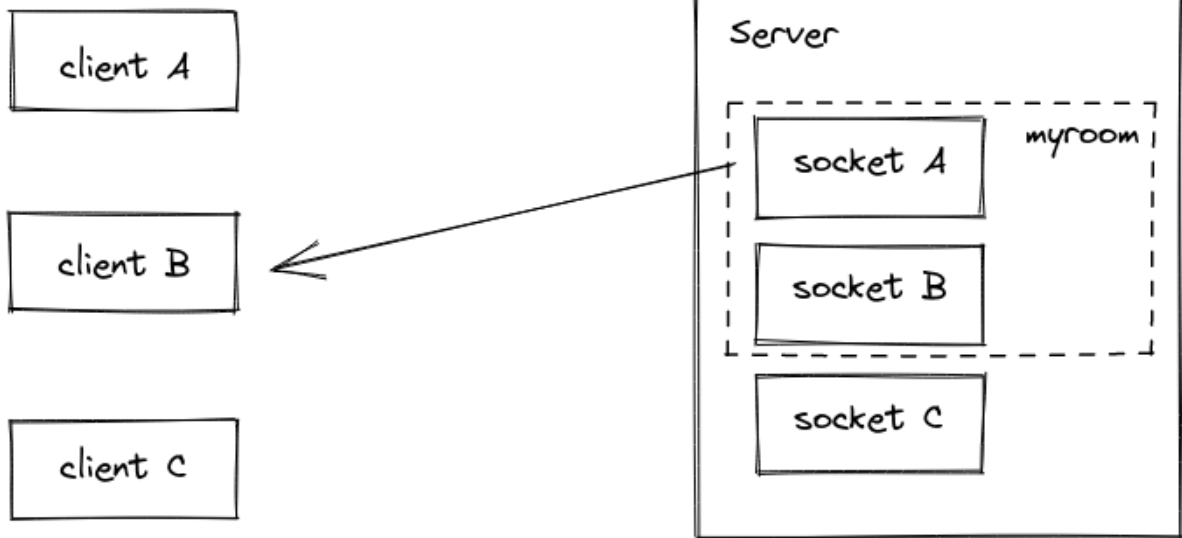
Socket.IO is a library that enables **low-latency**, **bidirectional** and **event-based** communication between a client and a server.



**Rooms in socket io-** A *room* is an arbitrary channel that sockets can `join` and `leave`. It can be used to broadcast events to a subset of clients:



In that case, every socket in the room **excluding** the sender will get the event.





# **REQUIREMENTS AND DESIGN**

## **1. Hardware Requirements**

### **For Server:**

- Active internet Connection
- Minimum 8 GB RAM
- Minimum 128GB SSD

### **For Client:**

- Active internet Connection

## **2. Software Requirements**

### **For Server:**

- Any Operating System
- Node JS
- nodemon
- Visual Studio Code-(IDE)

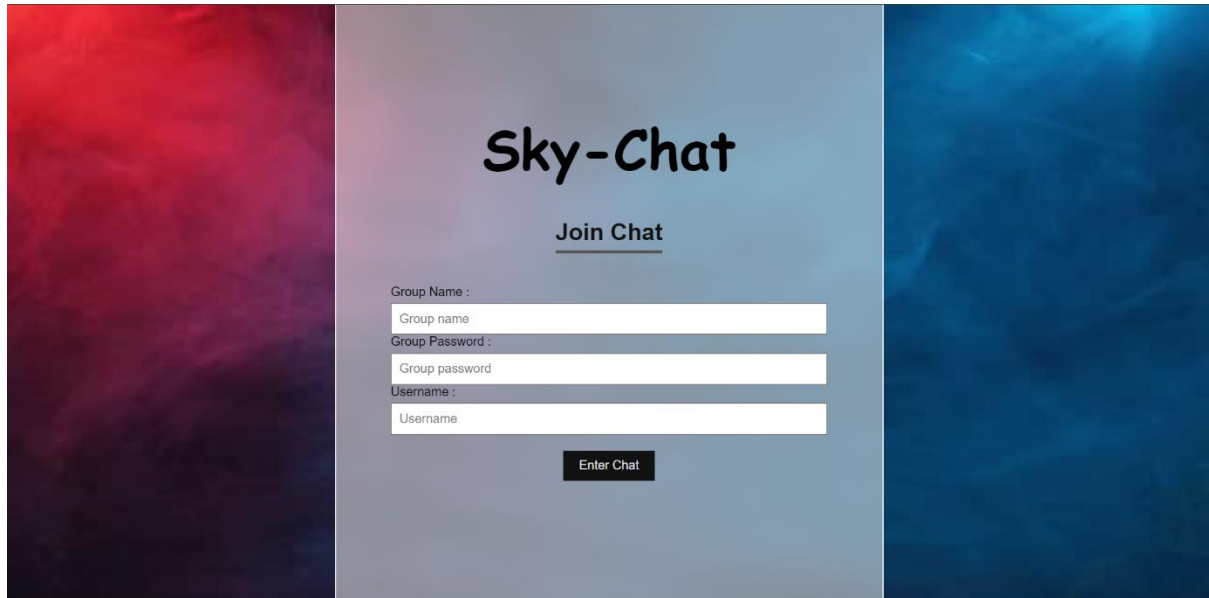
### **For Client:**

- Any Operating System
- Any Web Browser

## **3. Libraries**

- express
- nodemon
- path
- socket.io

## 4.Group Chat Login Page :-



The image shows the 'Join Chat' interface of Sky-Chat. It features a central grey panel with the title 'Sky-Chat' and a subtitle 'Join Chat'. Below the subtitle are three input fields: 'Group Name', 'Group Password', and 'Username'. Each field has a label above it and a placeholder text inside. A black 'Enter Chat' button is positioned below the input fields. The background consists of a red and purple gradient on the left and a blue gradient on the right.

**Sky-Chat**

Join Chat

Group Name :  
Group name

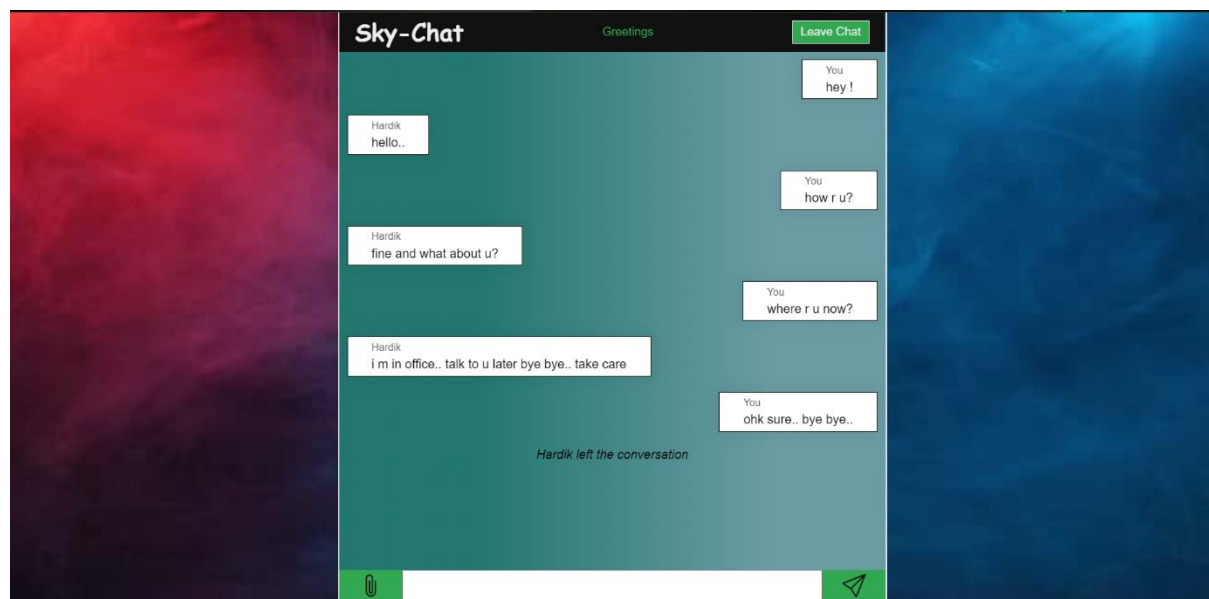
Group Password :  
Group password

Username :  
Username

Enter Chat

This is the interface which will open when the user log into the Sky-Chat by his/her login Credentials.

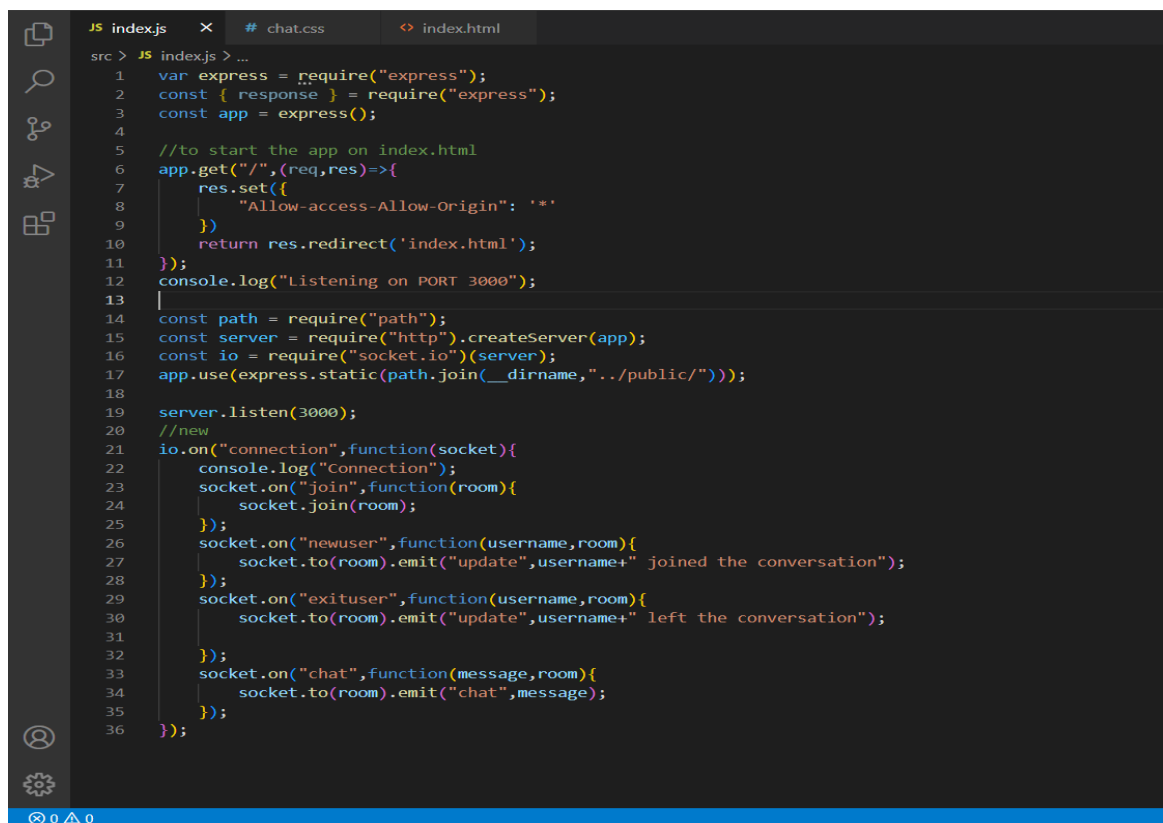
Here firstly user have to enter the group name followed by a group password and then a username.



Once the user enters the group he/she can wait for the other persons to join the room and when the new user joins the group the message in the group will come that the user has joined the conversation and after that the users can chat with each other share the multimedia content with each other and if the user want to exit the chat they can press the leave chat button on the top left corner of the window and when the user leave chat, then the message in the group will come that the user left the conversation in the above screenshot I have created the demo chat to display the app.

## CHAT SERVER ENGINE :-

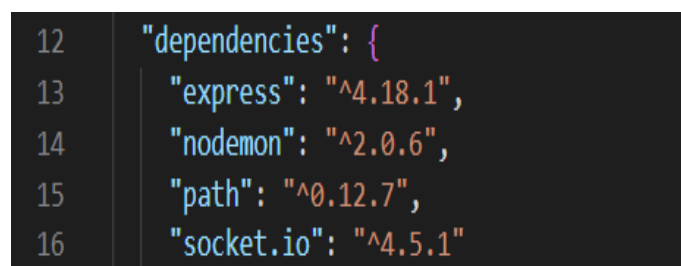
Chat server consist of the index.js file which is the node.js file which is run using nodemon index.js to start the server and the index.js file will somehow look like

A screenshot of a code editor with a dark theme. The editor has three tabs at the top: 'index.js' (selected), 'chat.css', and 'index.html'. The 'index.js' tab shows the following code:

```
src > JS index.js > ...
1  var express = require("express");
2  const { response } = require("express");
3  const app = express();
4
5  //to start the app on index.html
6  app.get("/",(req,res)=>{
7    res.set({
8      "Allow-access-Allow-Origin": '*'
9    })
10   return res.redirect('index.html');
11 });
12 console.log("Listening on PORT 3000");
13
14 const path = require("path");
15 const server = require("http").createServer(app);
16 const io = require("socket.io")(server);
17 app.use(express.static(path.join(__dirname,"../public/")));
18
19 server.listen(3000);
20 //new
21 io.on("connection",function(socket){
22   console.log("Connection");
23   socket.on("join",function(room){
24     socket.join(room);
25   });
26   socket.on("newuser",function(username,room){
27     socket.to(room).emit("update",username+" joined the conversation");
28   });
29   socket.on("exituser",function(username,room){
30     socket.to(room).emit("update",username+" left the conversation");
31   });
32   socket.on("chat",function(message,room){
33     socket.to(room).emit("chat",message);
34   });
35 });
36 });
```

The editor interface includes a sidebar on the left with icons for file explorer, search, source control, and a bottom status bar showing '0 0 0'.

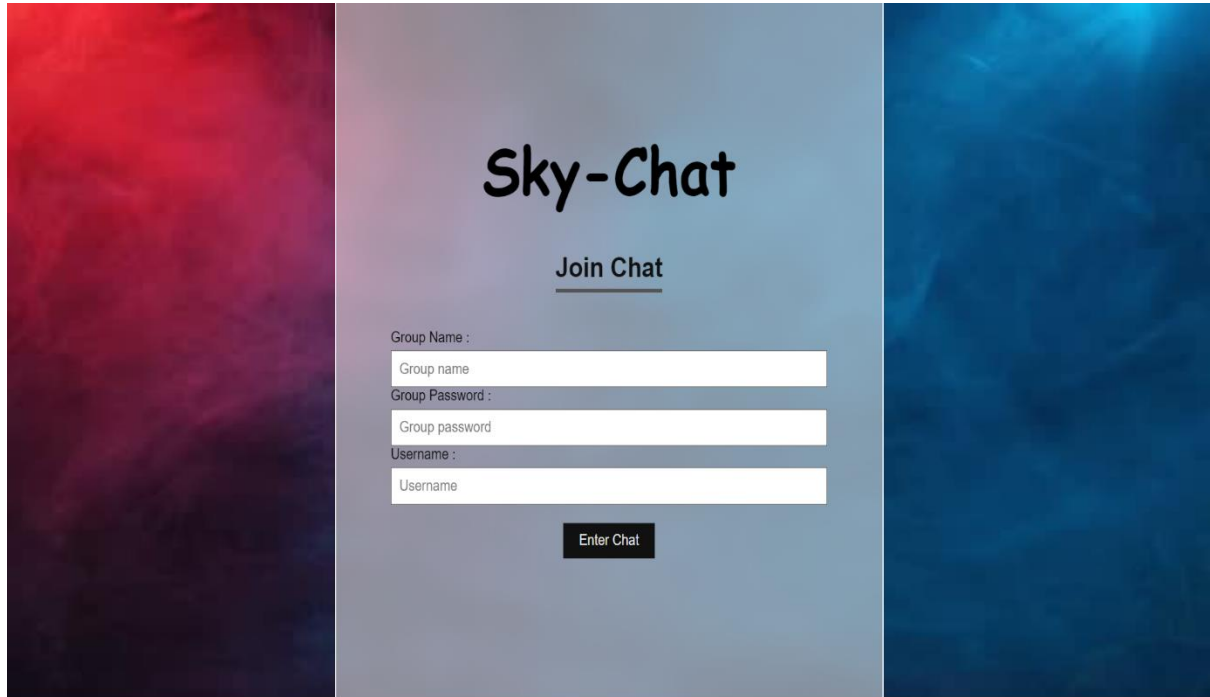
Other than this the dependencies that are required to run the server are :

A screenshot of a code editor showing a 'dependencies' object in a JSON file. The code is as follows:

```
12  "dependencies": {
13    "express": "^4.18.1",
14    "nodemon": "^2.0.6",
15    "path": "^0.12.7",
16    "socket.io": "^4.5.1"
```

# SCREENSHOT

## Login-Interface :-



**Sky-Chat**

Join Chat

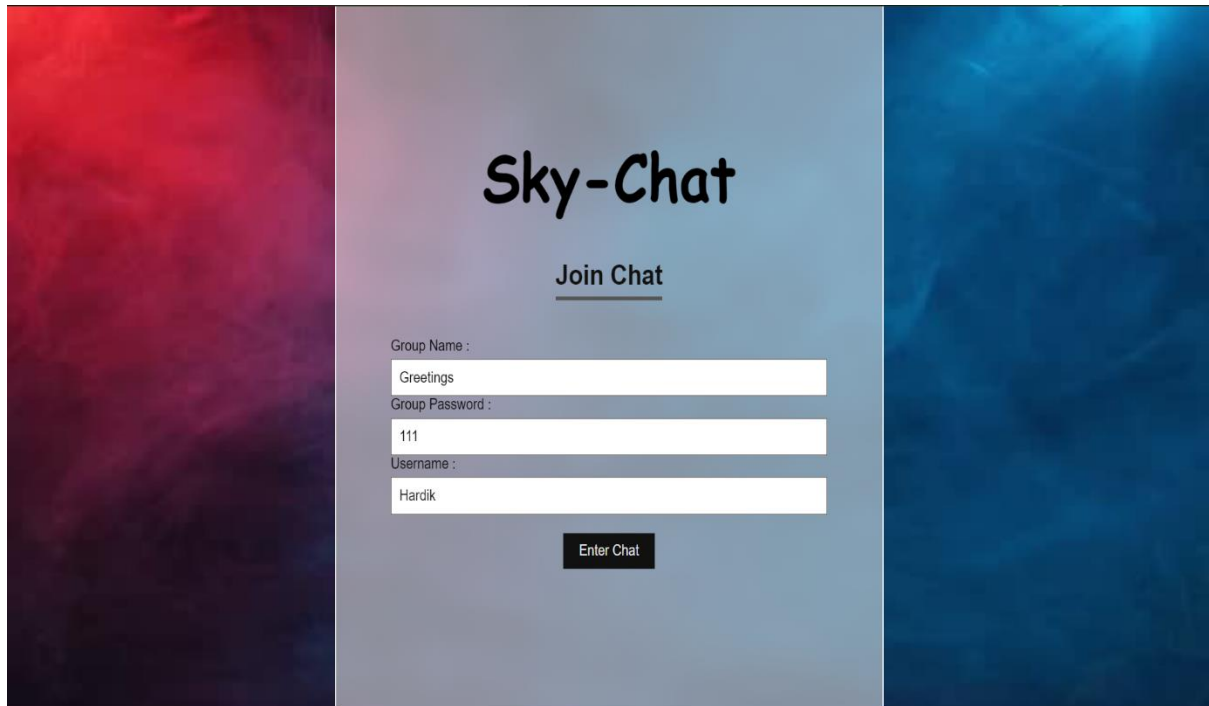
Group Name :

Group Password :

Username :

**Enter Chat**

## USER 1 :-



The image shows a web interface for 'Sky-Chat'. It has a central grey panel with a red gradient on the left and a blue gradient on the right. The title 'Sky-Chat' is at the top, followed by the link 'Join Chat'. Below are three input fields: 'Group Name' (containing 'Greetings'), 'Group Password' (containing '111'), and 'Username' (containing 'Hardik'). An 'Enter Chat' button is at the bottom.

Sky-Chat

[Join Chat](#)

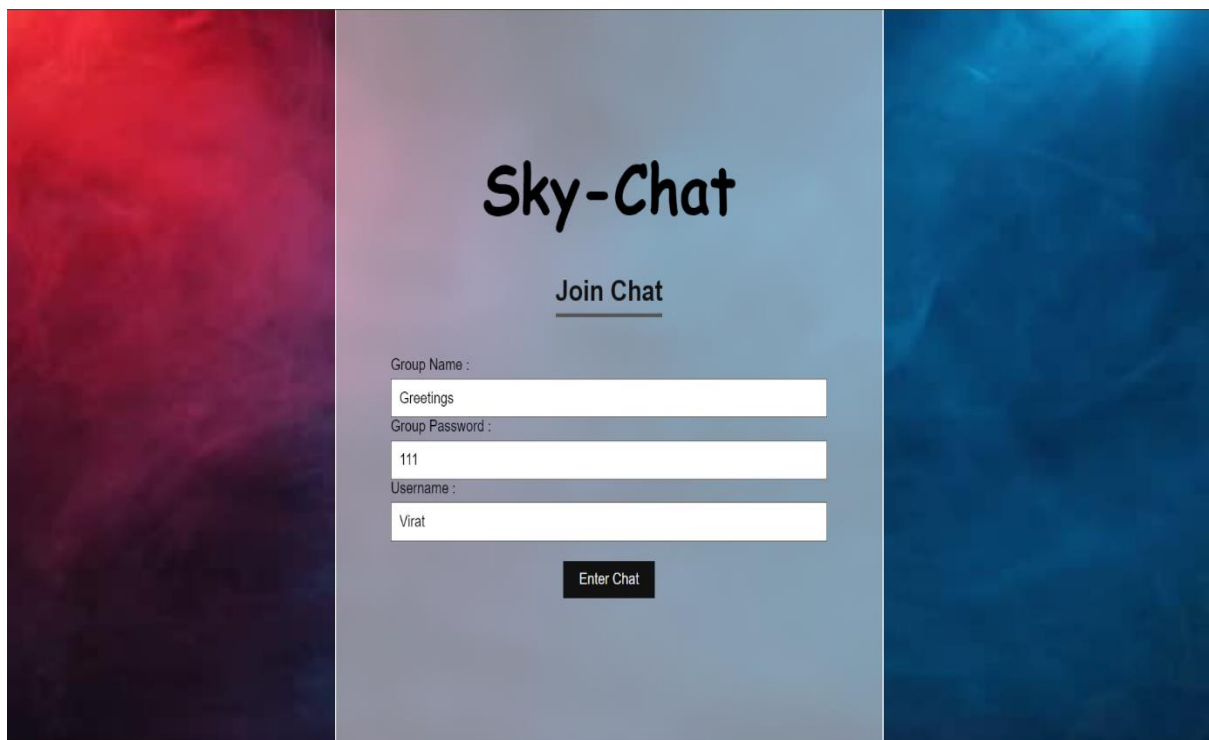
Group Name :  
Greetings

Group Password :  
111

Username :  
Hardik

Enter Chat

## USER 2 :-



The image shows the same 'Sky-Chat' web interface as above, but with the username 'Virat' entered in the 'Username' field.

Sky-Chat

[Join Chat](#)

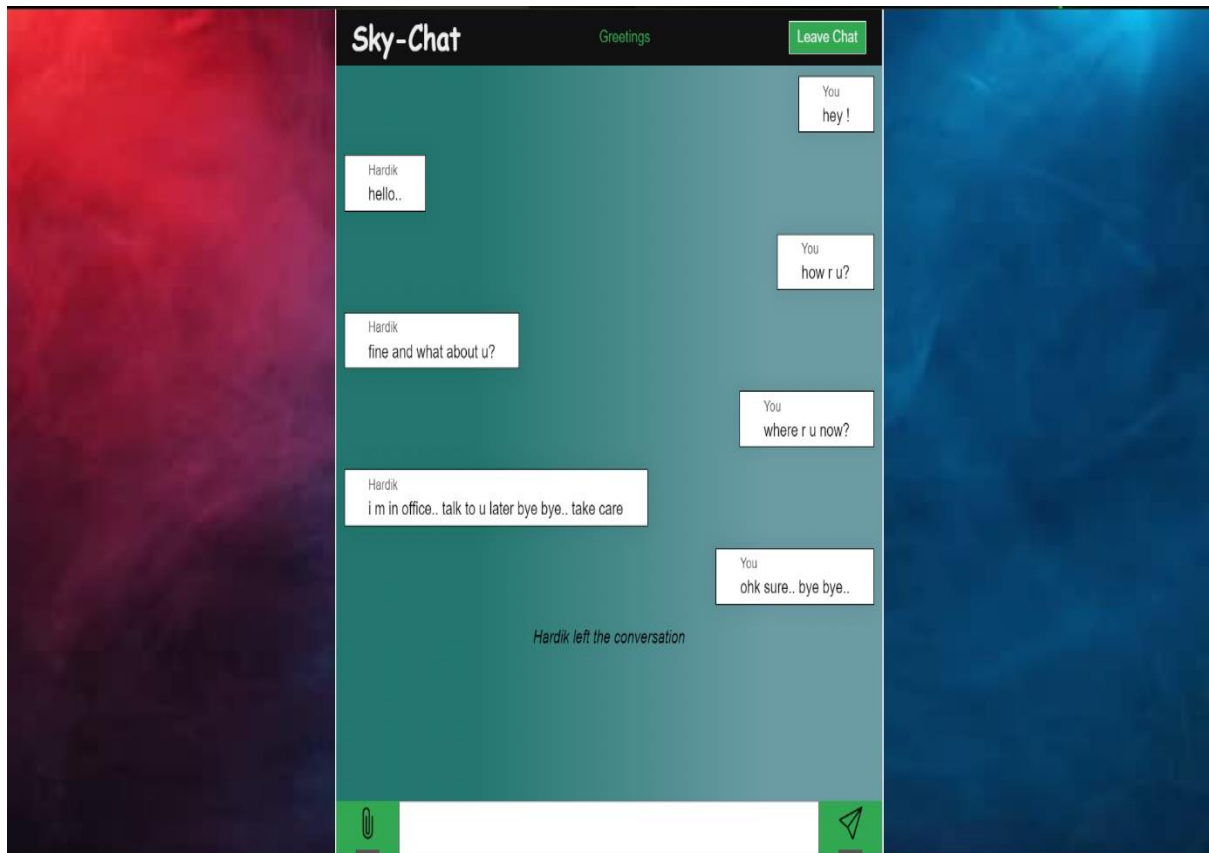
Group Name :  
Greetings

Group Password :  
111

Username :  
Virat

Enter Chat

## Chat – Screenshot :-



# **CONCLUSION**

There is always a room for improvements in any apps. Right now, we are just dealing with text communication and media sharing. There are several chat 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. 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:

- ✓ Voice Message
- ✓ Audio Call
- ✓ Video Call
- ✓ Group Call ,etc

## **REFERENCE**

ExpressJS : <https://www.javatpoint.com/expressjs-tutorial> ,  
<https://expressjs.com/en/guide/routing.html>

Npm : <https://www.npmjs.com/>

NodeJS : <https://nodejs.org/en/docs/>,  
<https://www.javatpoint.com/nodejs-tutorial>

Socket.IO : <https://socket.io/docs/v4/>