**CHAT APPLICATION USING SOCKET PROGRAMMING**

A COURSE PROJECT REPORT

By

**SURESH SUTHAR (RA2011003011323)**

**ADARSH (RA2011003011290)**

**SANSKAR GUPTA (RA2011003011308)**

**PUNEET AWASTHI (RA2011003011303)**

Under the guidance of   
**Mr. S. Saminathan***In partial fulfilment for the Course*

18CSC302J - COMPUTER NETWORKS

in the Department the of Computing Technologies



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Chenpalpattu District**

NOVEMBER 2022

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this mini project report "**CHAT APPLICATION USING SOCKET PROGRAMMING**" is the bonafide work of **SURESH SUTHAR (RA2011003011323), ADARSH (RA2011003011290), SANSKAR GUPTA (RA2011003011308)** and **PUNEET AWASTHI (RA2011003011303)** who carried out the project work under my supervision.

# SIGNATURE

**S. Saminathan**

# Assistant Professor

Department of C.Tech.

SRM Institute of Science and Technology

# ABSTRACT

This project briefly explains the concept of Socket Programming and shows its implementation with a use case scenario. Here the scenario is chat application made using Socket.io, React, JavaScript, Node.js, Express.js and Bulma.

A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time. With a chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.

Socket.IO is an event-driven library which enables real-time, bi-directional communication between web clients and servers. It consists of two parts: a client-side library that runs in the browser, and a server-side library for Node.js.

With this client and server communication through which the users are able to send/receive messages among the Internet users who are online or who were equally using the internet.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor, Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express our warmth of gratitude to our **Registrar, Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology), Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express our heartfelt thanks to **Chairperson, School of Computing**, **Dr. Revathi Venkataraman,** for imparting confidence to complete our course project.

We extend our gratitude to our **HoD,** **Dr. M. Pushpalatha, Professor and Head, Department of Computing Technologies** and our Departmental faculty members for their Support.

We wish to express our sincere thanks to **Course Audit Professor, Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and the **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty mentor, **Mr. S. Saminathan, Assistant Professor, Department of C.Tech,** for his assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, we thank the almighty for showering his blessings on us to complete our Course project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL. NO.** | **CHAPTER HEADING** | PAGE NO. |
| **1.** | **ABSTRACT** | **3** |
| **2.** | **ACKNOWLEDGEMENT** | **4** |
| **3.** | **INTRODUCTION** | **6** |
| **4.** | **DESIGN IMPLEMENTATION** | **8** |
| **5.** | **TESTING and ANALYSIS** | **9** |
| **6.** | **CONCLUSIONS & FUTURE ENHANCEMENT** | **10** |
| **7.** | **REFERENCES** | **11** |

**INTRODUCTION**

[Messaging](https://www.pubnub.com/learn/glossary/what-is-real-time-messaging/) has become a part of our everyday lives in part due to its convenience for[real-time chat communication](https://www.pubnub.com/learn/glossary/what-is-a-chat-api/) and simple-to-use functionality.

For instance, an iOS or text message on an iPhone or Android device from a friend, an email from a co-worker on Microsoft or Gmail, a team chat in a Slack or Microsoft Teams workspace, or even instant messaging through social media.

These messaging and real-time chat applications play an important role in how the world interacts today, due to their immediacy and vast capabilities.

A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time. With a [chat app](https://www.pubnub.com/use-case/in-app-chat/), users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.

This also keeps users conversing on your platform instead of looking elsewhere for a messaging solution. Whether it’s [private chat](https://www.pubnub.com/tutorials/getting-started-chat-sdk/), [group chat](https://www.pubnub.com/demos/group-chat-react-demo/), or [large-scale chat](https://www.pubnub.com/demos/virtual-events/), adding personalized chat features to your app can help ensure that your users have a memorable experience.

Socket.IO enables real-time bidirectional event-based communication. It works on every

platform, browser or device, focusing equally on reliability and speed. Socket.IO is built

on top of the WebSocket API (Client side) and Node.js. It is one of the most depended

upon library on npm (Node Package Manager).

Socket.IO enables real-time bidirectional event-based communication. It consists of:

* Node.js server (this repository)
* [Javascript client library](https://github.com/socketio/socket.io-client) for the browser (or a Node.js client)

Its main features are:

Reliability:

Connections are established even in the presence of:

* proxies and load balancers.
* personal firewall and antivirus software.

For this purpose, it relies on [Engine.IO](https://github.com/socketio/engine.io), which first establishes a long-polling connection, then tries to upgrade to better transports that are "tested" on the side, like WebSocket. Please see the [Goals](https://github.com/socketio/engine.io" \l "goals) section for more information.

Multiplexing Support:

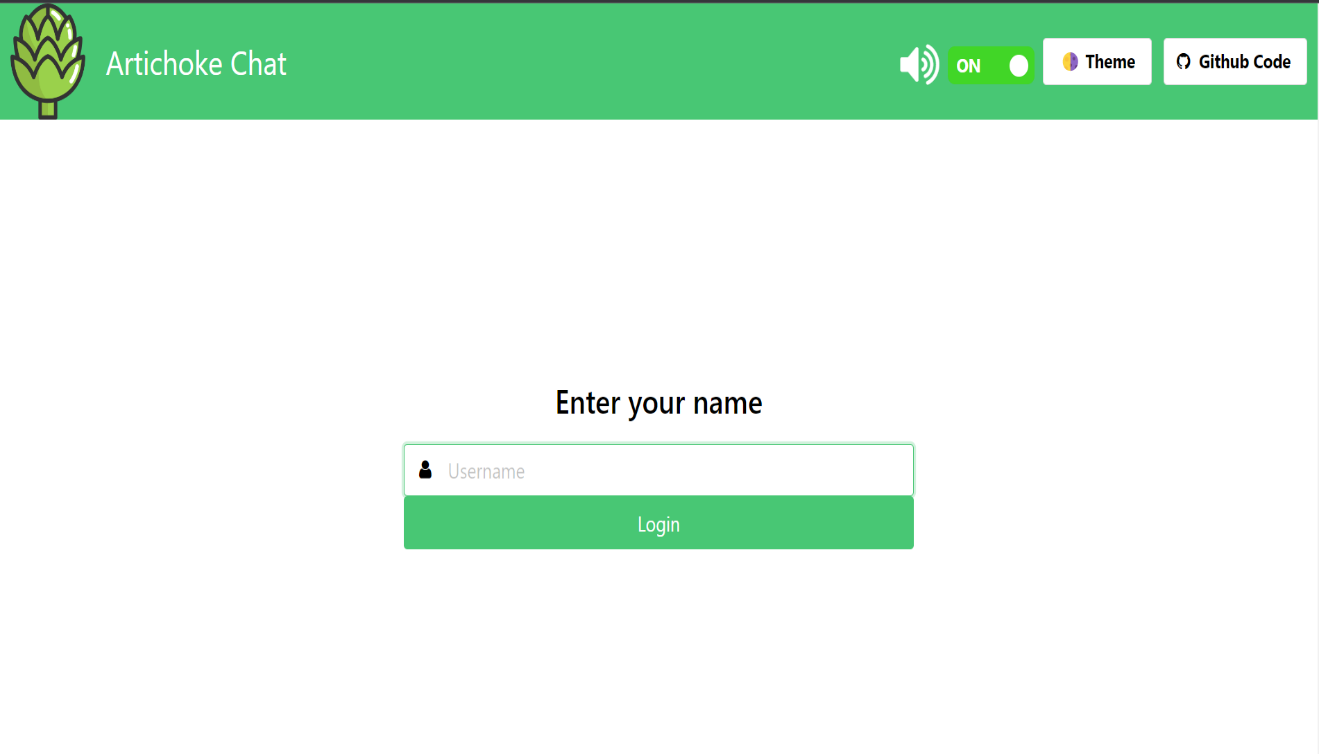
In order to create separation of concerns within your application (for example per module, or based on permissions), Socket.IO allows you to create several Namespaces, which will act as separate communication channels but will share the same underlying connection.

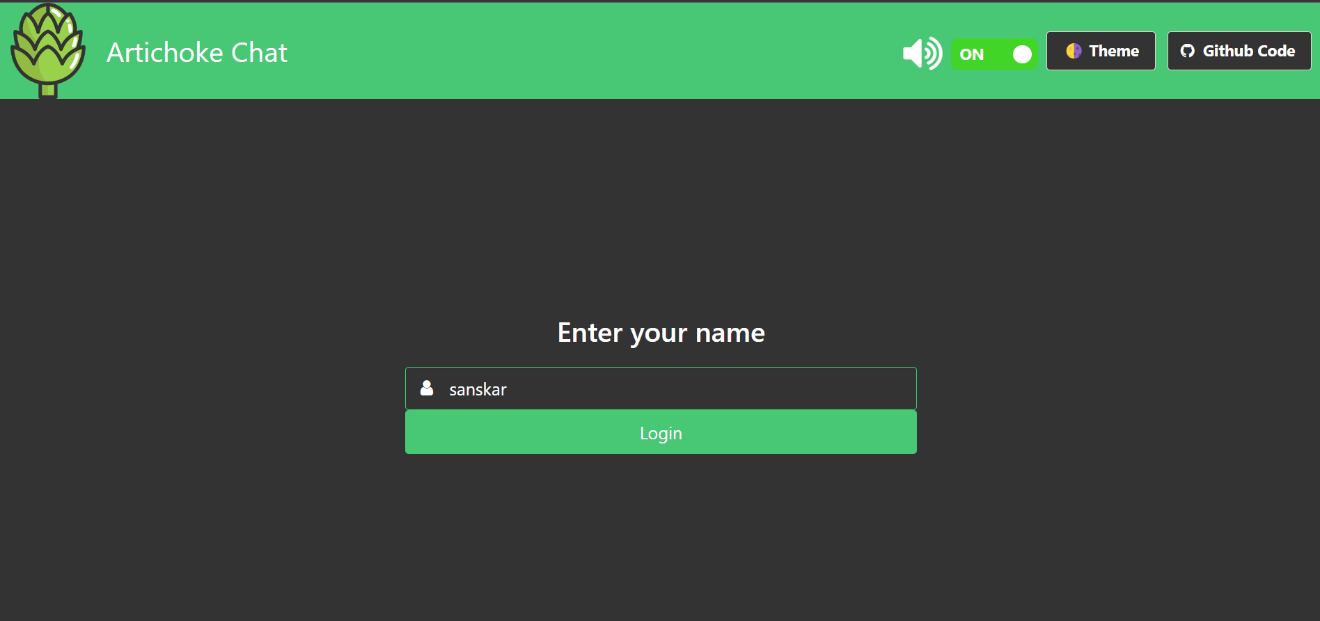
Room Support:

Within each Namespace, you can define arbitrary channels, called Rooms, that sockets can join and leave. You can then broadcast to any given room, reaching every socket that has joined it.

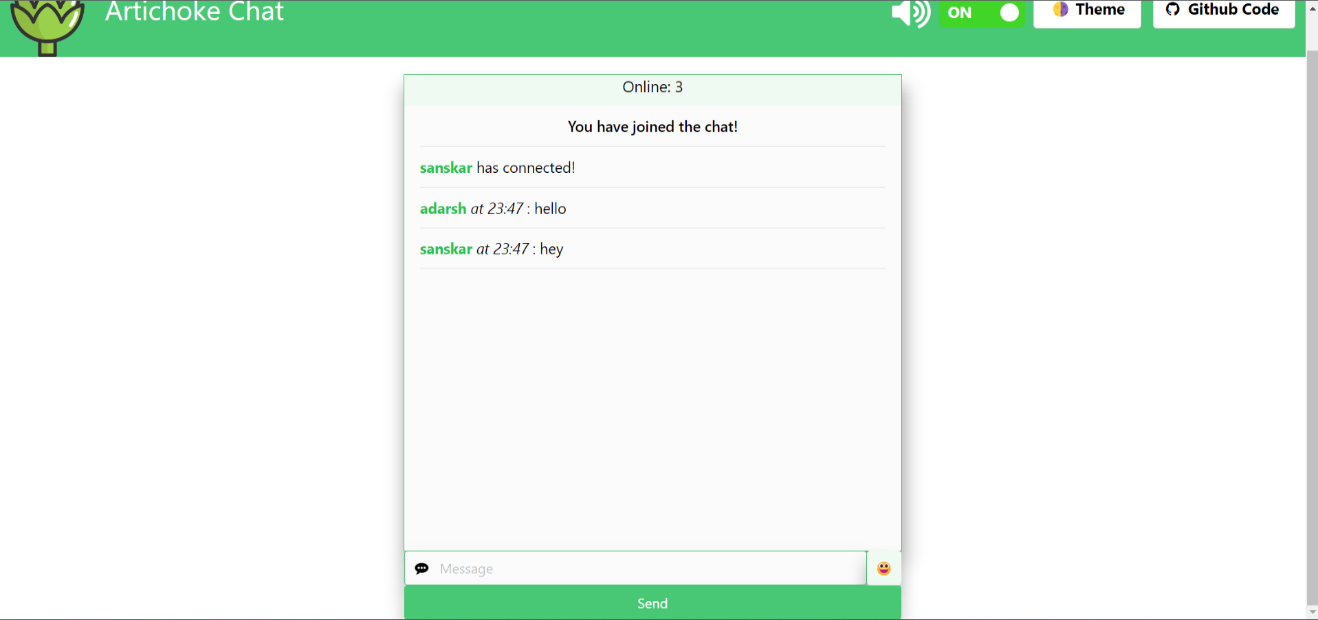
This is a useful feature to send notifications to a group of users, or to a given user connected on several devices for example.

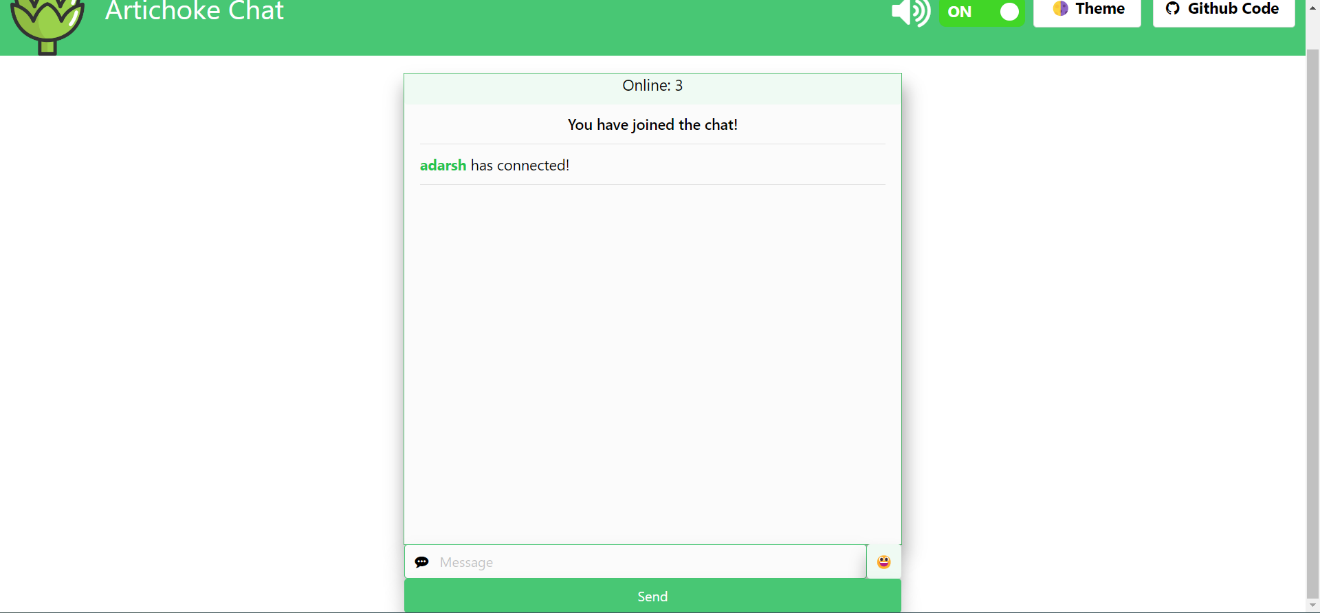
**DESIGN IMPLEMENTATION**

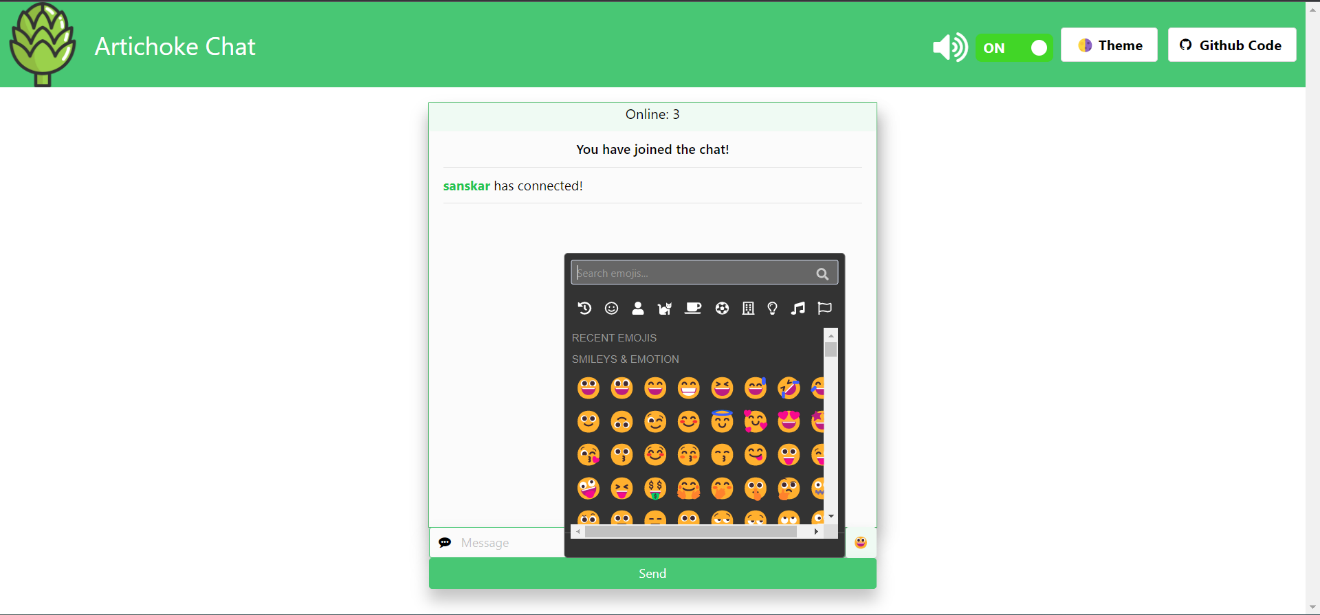
****

****

**TESTING and ANALYSIS**

****

****



**CONCLUSION and FUTURE ENHANCEMENT**

The chat app provides a better and more flexible chat system. Developed with the latest technology in the way of providing a reliable system. The main advantage of the system is instant messaging, real-world communication, added security, group chat, etc. This application may find the best demand in the market for most organizations that aim to have independent applications.

**Future Enhancements:**

* Extending this application by providing Authorisation service. Be able to save the credentials.
* Creating Database and maintaining users.
* Increasing the effectiveness of the application by providing Voice Chat.

**REFERENCES**

<https://github.com/socketio/socket.io>

<https://www.liveagent.com/customer-support-glossary/chatting-apps/>

<https://www.tutorialspoint.com/socket.io/socket.io_chat_application.htm>

<https://www.cometchat.com/tutorials/how-to-build-a-chat-app-with-socket-io-node-js>