**Web Chat**

Report submitted in partial fulfilment of the requirement for the degree of

 B. Tech

In

Computer Science &Engineering



Under the Supervision of By

**Arnav Gupta Shubham Prakash**

**Group G9**

Bhagwan Parshuram Institute of Technology

PSP-4, Sector-17, Rohini, Delhi - 89

June-July 2017

**DECLARATION**

This is to certify that Report entitled “Web Chat” which is submitted in partial fulfilment of the requirement for the award of degree B. Tech. in Computer Science Engineering to BPIT, GGSIP University, Dwarka, Delhi comprises only my original work and due acknowledgement has been made in the text to all other material used.

**Date: Name of Student**

**Acknowledgement**

I am highly indebted to Coding Blocks for their guidance and constant supervision for providing necessary information regarding the project, also their support in completing the project. I would like to express my gratitude towards Mr. Arnav Gupta for their kind co-operation and encouragement, which helps us in completion of the project.

I would also like to express our special thanks of gratitude to our teacher Ms. Palak Girdhar, who gave us the excellent opportunity to do this wonderful project and guided us throughout.

**Company Certificate**

**Training Coordinator Certificate**

This is to certify that Report entitled “Web Chat” which is submitted by Shubham Prakash in partial fulfilment of the requirement for the award of degree B. Tech in Computer Engineering to BPIT, GGSIP University, Dwarka, Delhi is a record of the candidate own work and the matter embodied in this report is adhered to the given format.

**Date: Coordinator**

LIST OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Contents** | **Page No.** |
| 1. | Abstract | 01 |
| 2. | Introduction | 02 |
| 3. | SRS | 03 |
| 4 | Diagrams | 04 |
| 5. | Technological Description | 07 |
| 6. | Process Selection | 08 |
| 7. | Working of Project | 13 |
| 8. | Future Scope | 16 |
| 9. | References | 17 |

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| S. No. | Figures & Diagrams | Page No. |
| 1. | Use Case Diagram | 05 |
| 2. | Entity Relationship Diagram | 06 |

**CHAPTER – 1**

1. **ABSTRACT**

A very simple cross- platform client-server chat application has been implemented in node.js with socket.io. Its design is described, limitations are discussed, improvements are proposed and a user manual is included.

Implementing a chat server application provides a good opportunity for a beginner to design and implement a network-based system. The design is done with CSS. It is implemented with node.js and socket.io, since is easy to program in, it precludes the need to deal with low-level memory management and includes powerful libraries for sockets and threads.

**CHAPTER – 2**

1. **INTRODUCTION**

Implementing a chat server application is one of the most popular network programming projects with newbie programmers. Tons of free source code is available on the web. Excellent highly configurable applications are available both as open source well as proprietary software. Some examples of open source Chat applications implemented in Java are Llama Chat, FreeCS, Chipchat and OpenCHAT. With little experience in network programming as well as a short duration for the project, my intension was not to match or improve the existing implementations but to implement a basic version on my own.

**CHAPTER – 3**

1. **System Requirement Specification**

The Application requires for the chat communications between people, so there will private web chat rooms based website in which two people can interact with each on the platform. When Chat Room is created, another member could join the room with given link, by providing name and e-mail address. There could be multiple chat rooms and many people chatting through them.

The USP of the system is that all the client application is based on JavaScript, which makes the system very fast a user friendly. Secondly the system will provide the user with easy to use and understand graphic interface. User can easily interact with the system with menus and text areas.

**CHAPTER – 4**

1. **DIAGRAMS**

**4.1 USE CASE DIAGRAM**

A use case diagram is a graphic depiction of the interactions among the elements of a system.

A [use case](http://searchsoftwarequality.techtarget.com/definition/use-case) is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service [Web site](http://searchsoa.techtarget.com/definition/Web-site). Use case diagrams are employed in [UML](http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language) (Unified Modelling Language), a standard notation for the modelling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and [debugging](http://searchsoftwarequality.techtarget.com/definition/debugging) a [software](http://searchsoa.techtarget.com/definition/software) product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalogue updating, payment processing, and customer relations. A use case diagram contains four components.

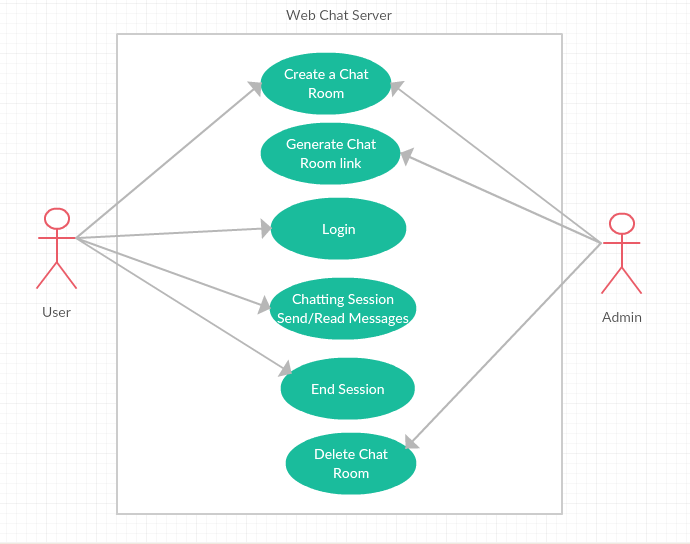


Figure 1.Use Case Diagram

* 1. **ENTITY RELATIONSHIP DIAGRAM**

Definition - What does Entity-Relationship Diagram (ERD) mean?

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

* Entities
* Relationships
* Attributes

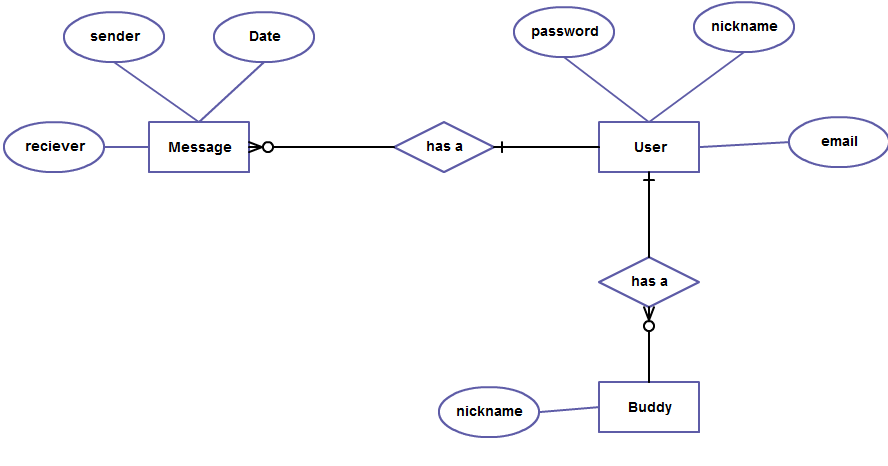


Figure 2.Entity Relationship Diagram

**CHAPTER - 5**

1. **TECHNOLOGY DESCRIPTION**

Writing a chat application with popular web applications stacks like LAMP (PHP) has traditionally been very hard. It involves polling the server for changes, keeping track of timestamps, and it’s a lot slower than it should be.

Sockets have traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.

This means that the server can *push* messages to clients. Whenever we write a chat message, the idea is that the server will get it and push it to all other connected clients.

The first goal is to setup a simple HTML webpage that serves out a form and a list of messages. We’re going to use the Node.JS web framework express to this end.

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system) for executing JavaScript code [server-side](https://en.wikipedia.org/wiki/Server-side). Historically, JavaScript was used primarily for [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting), in which scripts written in JavaScript are embedded in a webpage's HTML, to be run client-side by a JavaScript engine in the user's web browser. Node.js enables JavaScript to be used for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting), and runs scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js has become one of the foundational elements of the "JavaScript everywhere" paradigm, allowing [web application](https://en.wikipedia.org/wiki/Web_application) development to unify around a single programming language, rather than rely on a different language for writing server side scripts.

**CHAPTER – 6**

1. **PROCESS SELECTION (Coding)**

The Software application runs through html files and designed by CSS. The node.js and socket.io with the platform JavaScript runs on the back-end.

The application has Server which is hosted by the administrator and used by the Client on the local host link. Part of the html code is attached for the reference-

Home.html & Chat.html (A part of the code for the representative purpose)

Home.html:-

<!DOCTYPE **html**>  
  
*<!--   
 This file is rendered by express.js, when the home url of the chat is opened in a browser.  
 It doesn't do much, except showing the create button for the chat.  
 -->*<**html**>  
  
<**head**>  
  
 <**title**>Create a new chat room! </**title**>  
  
 <**link type="text/css" rel="stylesheet" href="css/stylesheet.css"** />  
 <**link href="http://fonts.googleapis.com/css?family=Open+Sans Condensed:300italic,300,700" rel="stylesheet" type="text/css"**>  
  
</**head**>  
  
<**body**>  
  
 <**header class="homebanner"**>  
 <**h6**>  
 Copyright © Shubham Prakash  
 </**h6**>  
  
 <**h1 class="homebannertext"**>  
 Web Chat <**span**>: Connecting People</**span**>  
 </**h1**>  
  
 </**header**>  
  
 <**section**>  
  
 <**div class="homesection"**>  
  
 <**a title="Create" href="/create" id="create"**>  
 <**div id="createbutton"**>  
 <**div id="little"**>Create a Private</**div**>  
 <**div id="big"**>CHAT ROOM!</**div**>  
 </**div**>  
 </**a**>  
  
 </**div**>  
  
 </**section**>  
  
  
  
</**body**>  
</**html**>

Chat.html:-

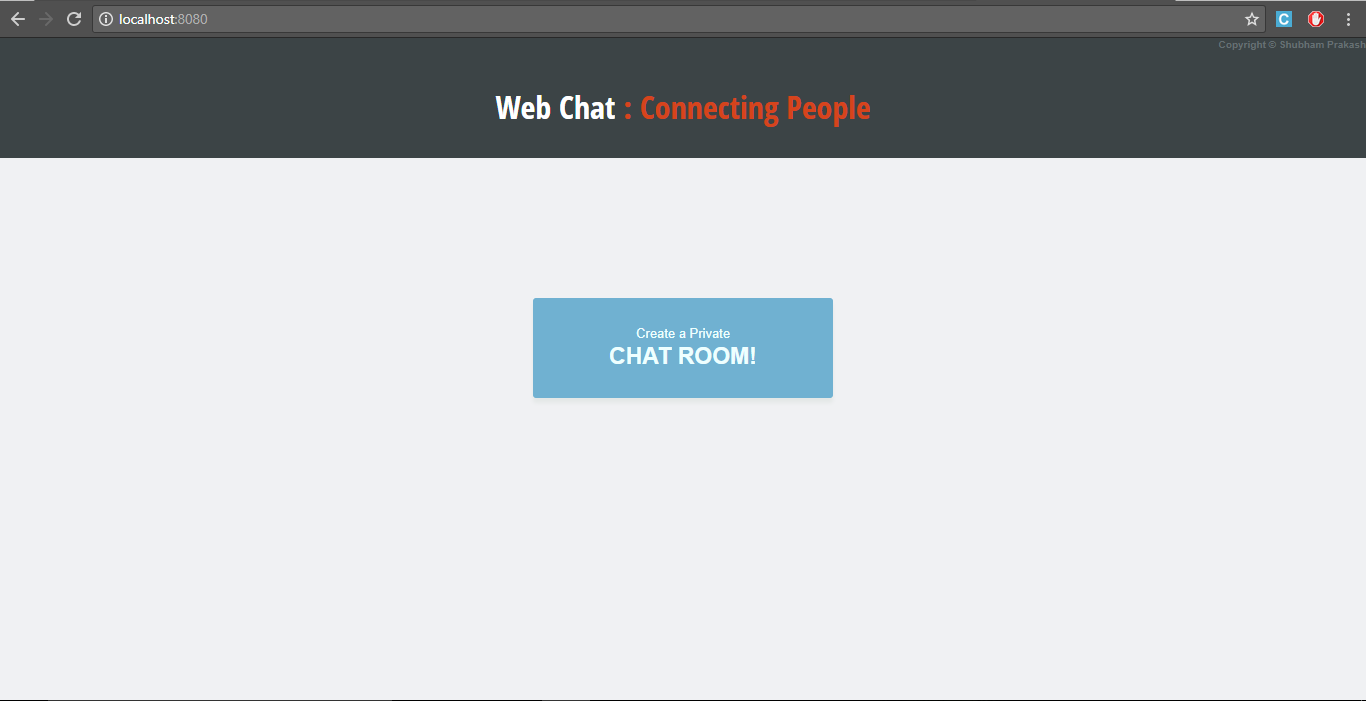
<!DOCTYPE **html**>  
*<!--  
 This file is rendered by express.js, when the rurl /chat/123456 is visited in a browser.  
 It includes jQuery, socket.io.js (it is automatically served by the socket.io library),   
 and a few more JavaScript files that you should check out.  
-->*<**html**>  
  
<**head**>  
  
 <**title**>Private chat room | Web Chat</**title**>  
  
 <**link type="text/css" rel="stylesheet" href="../css/stylesheet.css"** />  
 <**link href="http://fonts.googleapis.com/css?family=Open+Sans Condensed:300italic,300,700" rel="stylesheet" type="text/css"**>  
  
  
</**head**>  
  
<**body**>  
  
 <**header class="banner"**>  
 <**h6**>  
 Copyright © Shubham Prakash  
 </**h6**>  
  
 <**h1 class="bannertext"**>  
 Web Chat  
 </**h1**>  
  
 </**header**>  
  
 <**section class="section"**>  
  
 *<!-- These elements are displayed as white info cards in the middle of the screen -->* <**div class="connected"**>  
  
 <**img src="../img/unnamed.jpg" id="creatorImage"** />  
  
 <**div class="infoConnected"**>  
 <**h2**>Who are you?</**h2**>  
 <**br**/>  
  
 <**form class="loginForm"**>  
 <**input type="text" id="yourName" placeholder="Your nick name"** /><**br**/>  
 <**input type="text" id="yourEmail" placeholder="Your email address"** /><**br**/>  
 <**input type="submit" id="yourEnter" value="ENTER"** />  
 </**form**>  
  
 </**div**>  
  
 </**div**>  
  
 <**div class="personinside"**>  
  
 <**img src="../img/unnamed.jpg" id="ownerImage"** />  
  
 <**div class="infoInside"**>  
 <**h2**>Chat with <**span class="nickname-chat"**></**span**></**h2**>  
 <**br**/>  
  
 <**form class="loginForm"**>  
 <**input type="text" id="hisName" placeholder="Your nick name"** /><**br**/>  
 <**input type="text" id="hisEmail" placeholder="Your email address"** /><**br**/>  
 <**input type="submit" id="hisEnter" value="CHAT"** />  
 </**form**>  
  
 </**div**>  
  
 </**div**>  
  
 <**div class="invite-textfield"**>  
  
 <**h2**>Oops, there are no other people in this chat!</**h2**>  
 <**h5**>Invite a friend by sending them this URL</**h5**>  
  
 <**div class="link"**>  
 <**a title="Invite a friend" href="" id="link"**></**a**>  
 </**div**>  
  
 </**div**>  
  
 <**div class="left"**>  
  
 <**img src="../img/unnamed.jpg" id="leftImage"** />  
  
 <**div class="info"**>  
 <**h2**><**span class="nickname-left"**></**span**> has left this chat.</**h2**>  
 <**h5**>Invite somebody else by sending them this page.</**h5**>  
 </**div**>  
  
 </**div**>  
  
 <**div class="toomanypeople"**>  
  
 <**h2**>Oops, you can not join this chat!</**h2**>  
 <**h5**>There are already two people in it. Would you like to create a <**a title="New Room" href="/create" id="room"**>new room</**a**>?</**h5**>  
  
 </**div**>  
  
 <**div class="nomessages"**>  
  
 <**img src="../img/unnamed.jpg" id="noMessagesImage"** />  
  
 <**div class="info"**>  
 <**h2**>You are chatting with <**span class="nickname-chat"**></**span**>.</**h2**>  
 <**h5**>Send them a message from the form below!</**h5**>  
 </**div**>  
  
 </**div**>  
  
 <**div class="chatscreen"**>  
  
 <**ul class="chats"**>  
 *<!-- The chat messages will go here -->* </**ul**>  
  
 </**div**>  
  
 </**section**>  
  
 <**footer**>  
  
 <**form id="chatform"**>  
  
 <**textarea id="message" placeholder="Write something.."**></**textarea**>  
 <**input type="submit" id="submit" value="SEND"**/>  
  
 </**form**>  
  
 </**footer**>  
  
 <**script src="http://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js"**></**script**>  
 <**script src="../js/moment.min.js"**></**script**>  
 <**script src="/socket.io/socket.io.js"**></**script**>  
 <**script src="../js/chat.js"**></**script**>  
  
</**body**>  
  
</**html**>

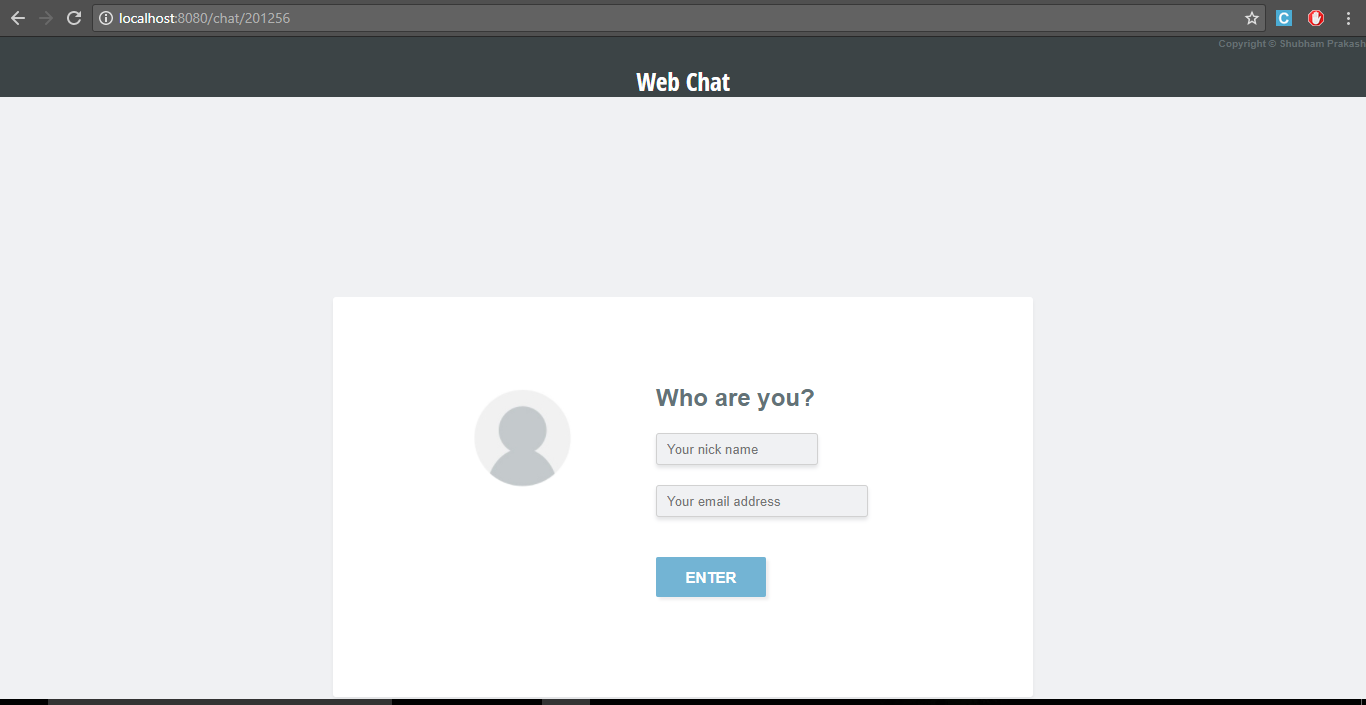
**CHAPTER – 7**

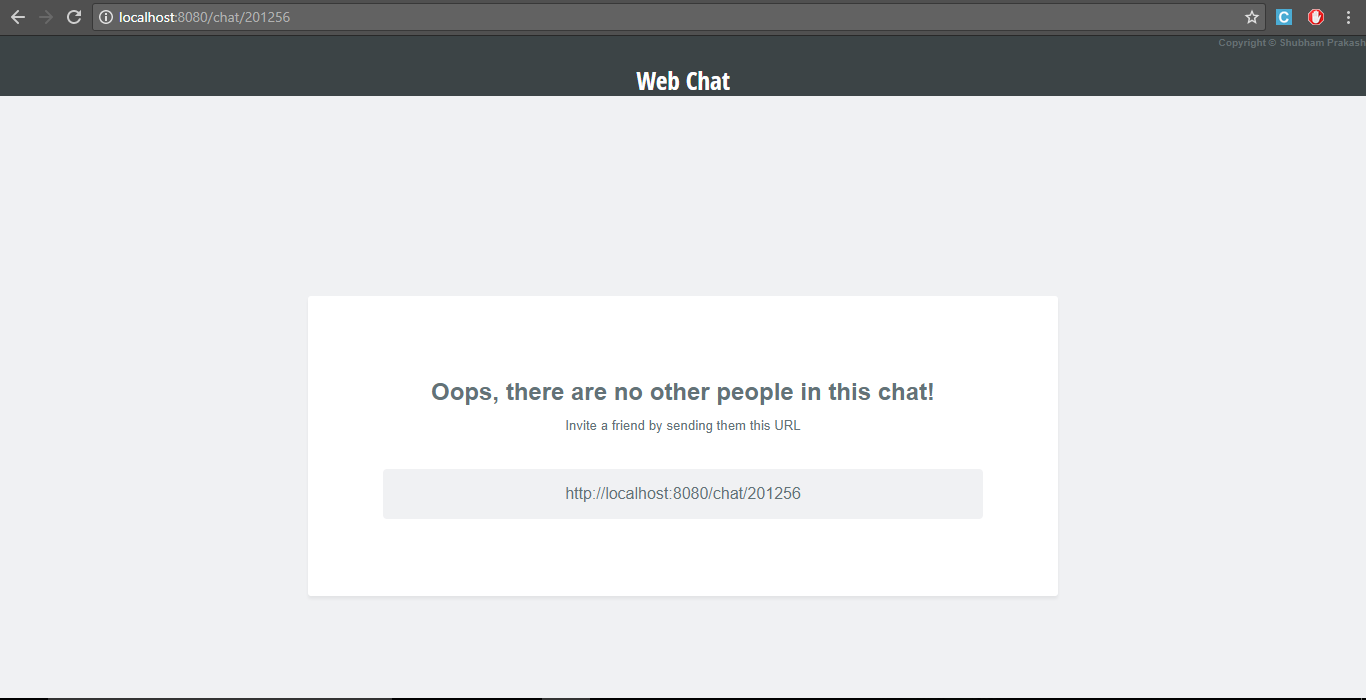
1. **WORKING OF PROJECT (Results)**

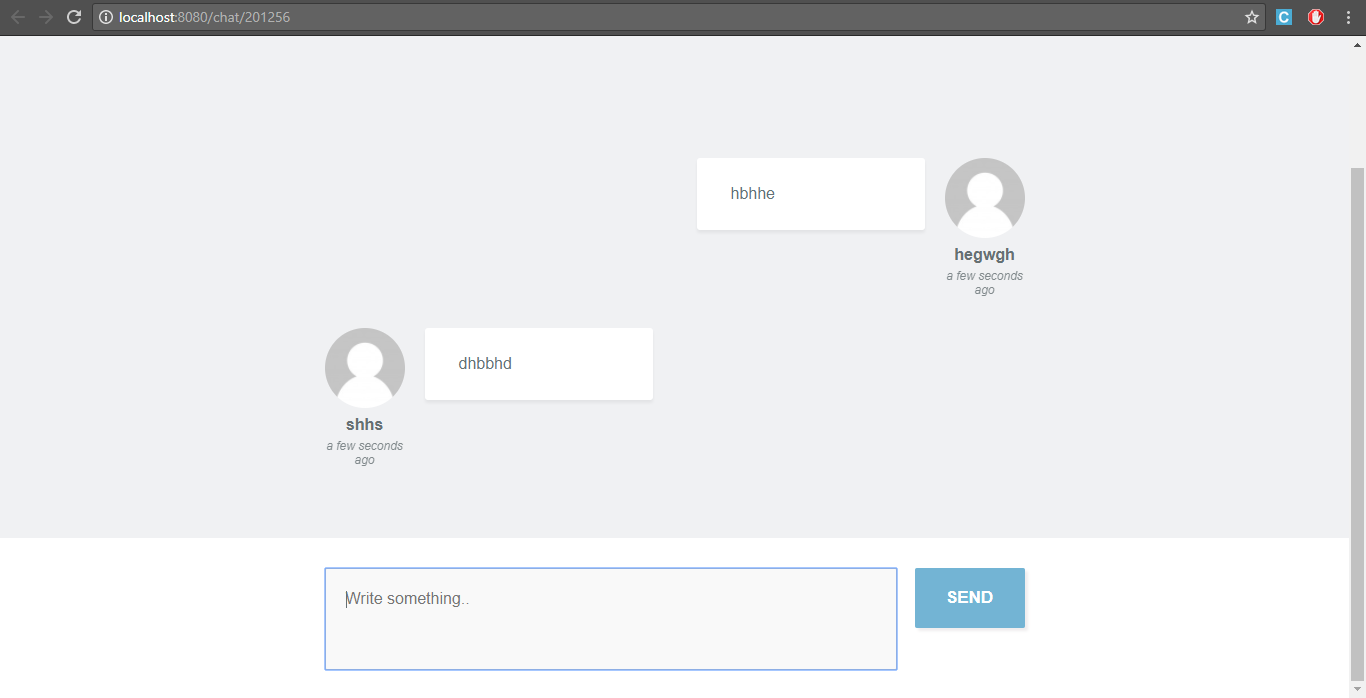
The main thread opens a server socket on the local inet address and port 8080, which has been arbitrarily chosen and hard coded. It then waits for clients to connect to it. When a client connects to the server, it creates a separate thread as well as a User object dedicated to it. The messages going from a client to the server are strings.

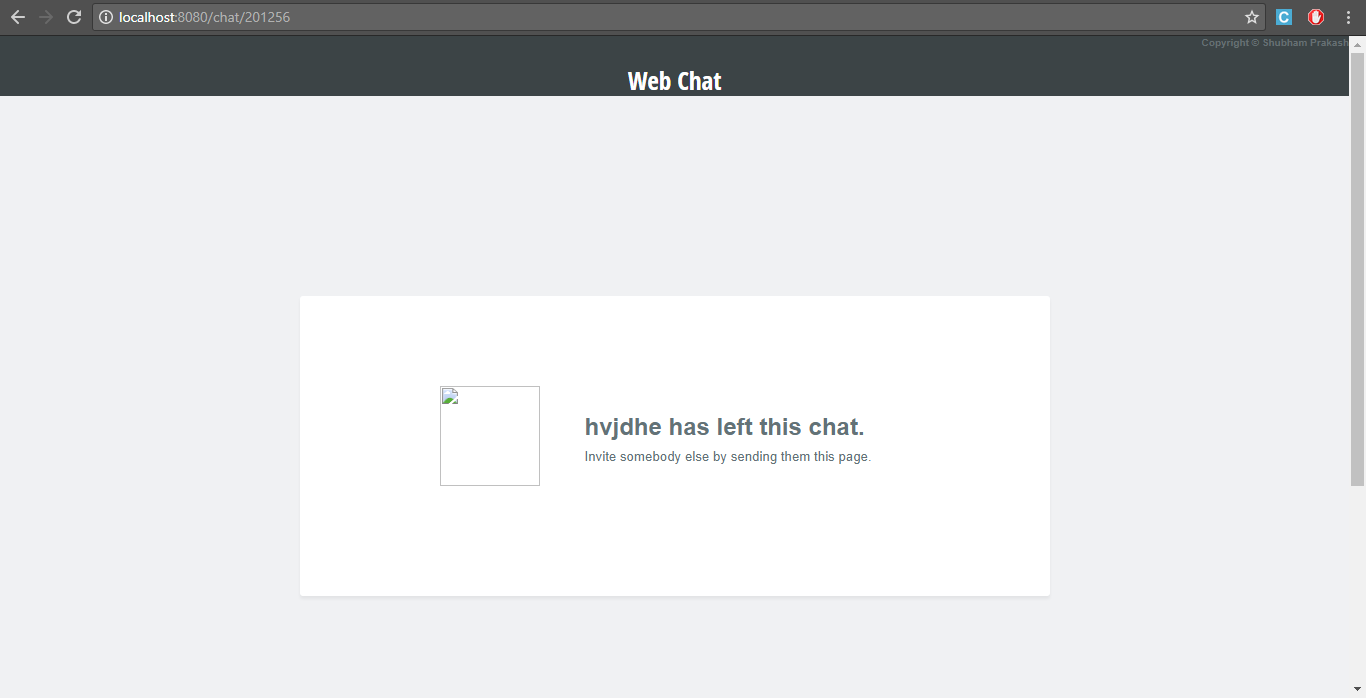
The pictures of working of the project are:-

Main Screen of the Program-

After we go to create the chat room, we have to fill the form with name and e-mail id-

After the creation of room and id, a link will be generated this can be provided to other member to chat-

While the people are chatting-

If a user leaves the chat in between, then the following message will be generated-

**CHAPTER - 8**

1. **FUTURE SCOPE**

A database of users containing username and password can be coupled with the existing design to maintain user accounts. An improved version can include multiple servers, serving different geographical locations, while talking to each other. This will preclude messages between clients located close to each other being routed through a server located in a far off location, thus decreasing the delay.

The Interoperability framework provides a high-level context and structure within which interoperability discussions can occur. It identifies key categories and issues that need to be addressed. It will not provide reference architecture and a high-level model that identifies the primary systems involved in the future “Smart Grid” along with key system boundaries, information exchanges and interactions. Such architecture ad model is needed in the future. The architecture would be based on the foundation that distributed generation, demand response and transmission /distribution technologies will become automated through the use of modern large-scale, distributed computing technology.

Here are some ideas to improve the application:

* Connecting to a database such that user’s information can be stored
* Don’t send the same message to the user that sent it himself. Instead, append the message directly as soon as he presses enter.
* Add “{user} is typing” functionality
* A facility for Group Chat
* Help and Feedback Sections

**CHAPTER - 9**

1. **REFERENCES**

**For Links:**  
<http://www.programwitherik.com/getting-started-with-socket-io-node-js-and-express/>

<https://socket.io/get-started/chat/>

<http://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js/>

**For books:**

Socket.IO Real-Time Web Application Development

Book by Rohit Rai

**For Journals:**

# [Slashie's Journal](https://blog.slashie.net/)

## Tales of a Game Developer