VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ONLINE MESSAGING WEBSITE CHATSTRIVE

Done by

19BCE0265 - Ayush Gupta

19BCE2146 - Tanmay Kumar

19BCE2440 - Ria Arun

19BCE2459 - Esha Jawaharlal

19BCE2486 - Medha Tripathi

For the course

Course Code: CSE3002

Course Name: Internet And Web Programming

FALL - 2021

TABLE OF CONTENTS

ABSTRACT

In 2021, where the world is in the midst of a technological revolution catalyzed by a pandemic, everything has shifted online. From our everyday mundane tasks of grocery shopping, to entertainment to even education, everything has found an online alternative for its offline counterpart. But one of the major things that has been introduced to digitalization way before anything else is communication in the form of texting and messaging, which has inspired us to come up with this project. In the heat of this global digitalization, along with the many perks of it all have come many cons in the forms of loss of privacy and malicious online activity. Unsafe communication can lead to a number of losses ranging from misconduct to identity theft, which can be very harmful to individuals. ChatStrive is a website that is going to offer services such as instant messaging, group chatting, file sharing and profile creation for every user on a very safe platform. It is a highly encrypted website for ensuring secure communication among individuals who can sign up to the website and avail its services.

INTRODUCTION

With the growth of Information Technology, communication has become easier than before. There are applications that help in the process of communication by transferring messages, images and files. Many such applications exist to serve as a communication platform to the vast population. During this pandemic, communication through these applications aided people to stay in contact with their close relations. This project aims to create an application that provides its users a safe and secure platform to communicate and stay in touch. Privacy and confidentiality of the user's of this application will be maintained.

This project is developed using HTML, CSS, React.js, Node.js and MongoDB. HTML is the standard markup language for documents designed to be built in a web browser. CSS is used for styling. React is used to create easy user interfaces. Node.js is a software platform that is used to create servers using event-driven programming. It runs Javascript on the server.

Features

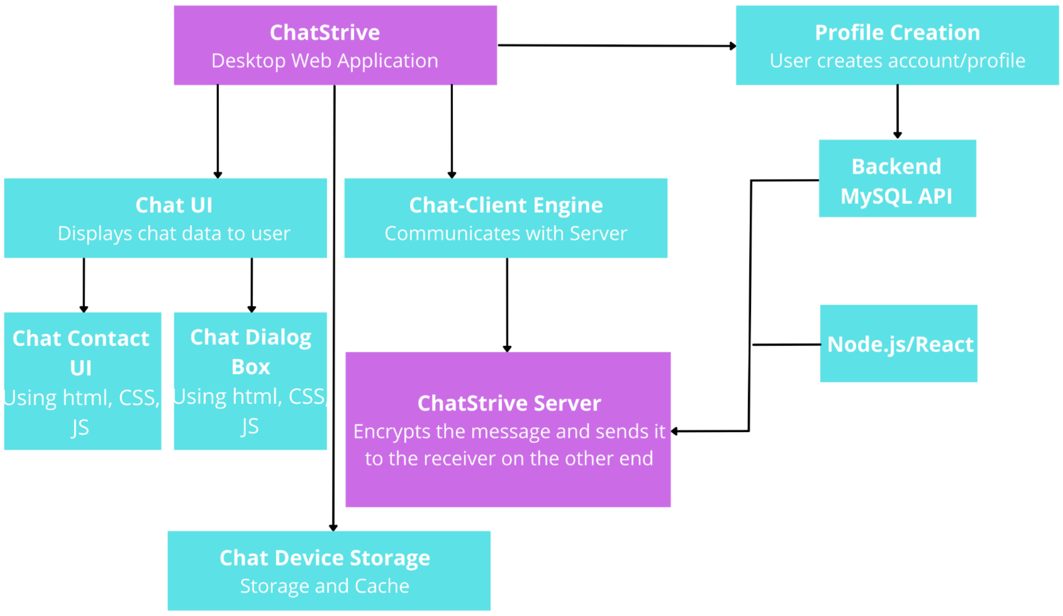
The features present in this applications are :

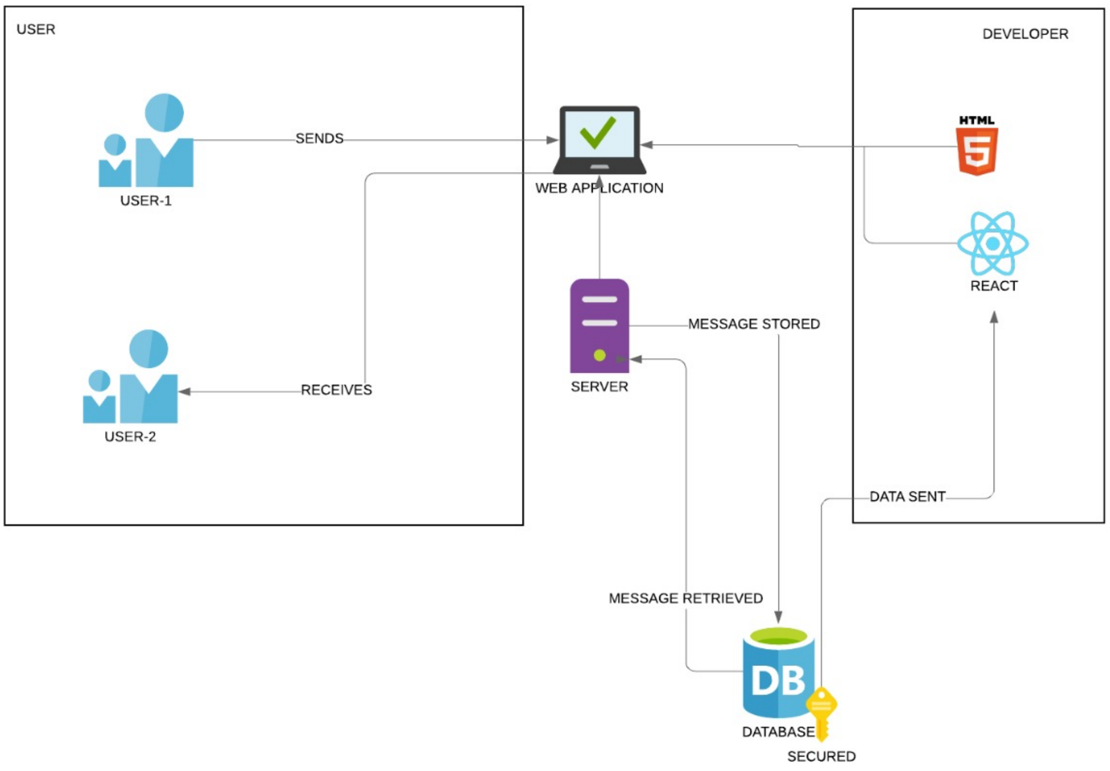
- Homepage : The main webpage.
- Sign-Up Page : This page is created to allow new users to register to the platform.
- Sign-In Page : This page ensures that the registered user has access to the platform, it is linked to a database which contains the user id and password of the registered users. Hence, allowing access.
- Chat window : The main window where the messages will be sent and received.
- Contact Us : This page contains a form that the user can fill in case of any queries or suggestions.

Scope

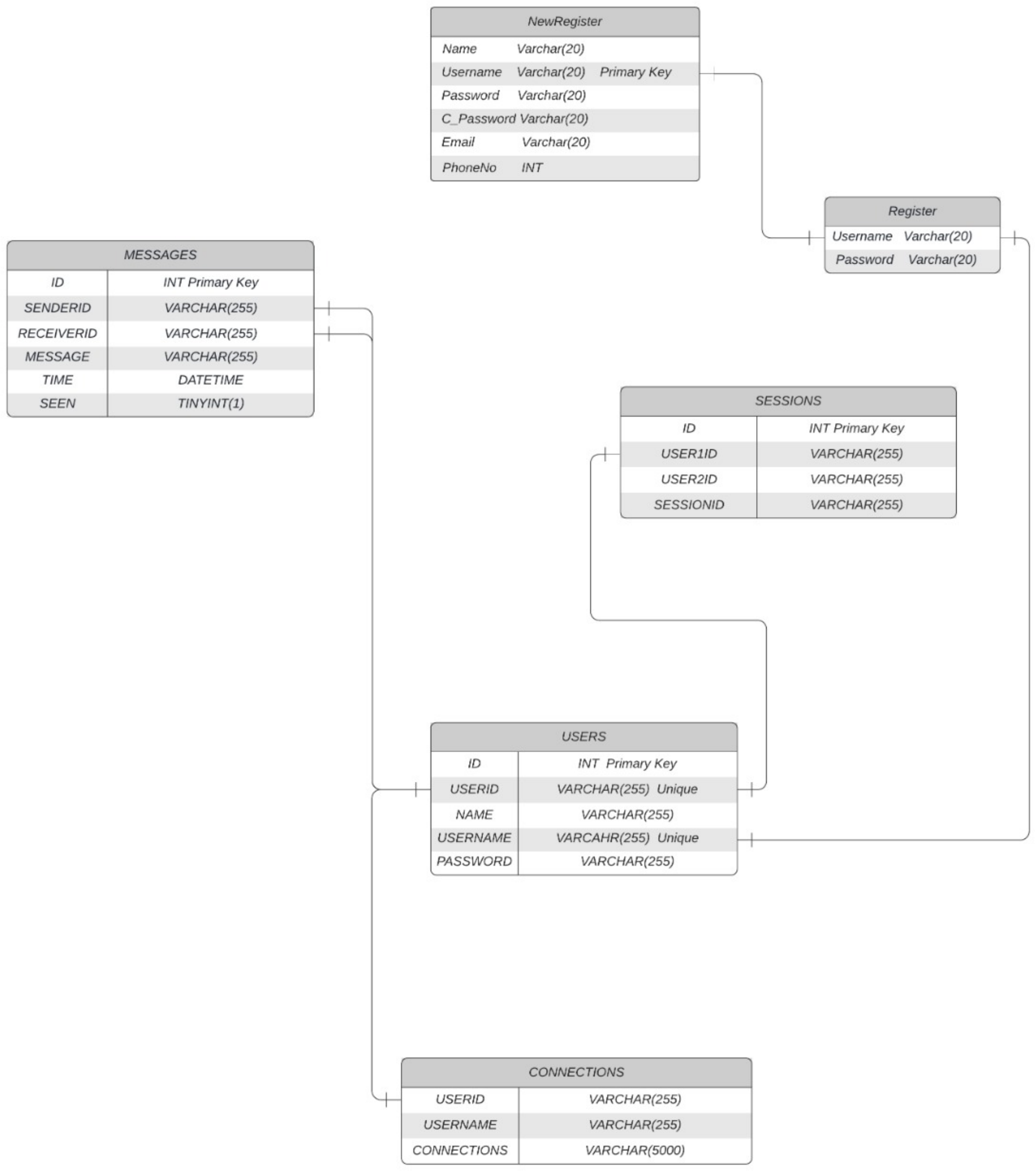
1. The design and building of this project is aimed at creating a web-based secure chat application system.
2. This system is built using React.js and Node.js .
3. Database of this system is implemented using MongoDB.

ARCHITECTURE DIAGRAM





DATABASE DESIGN



MODULE DESCRIPTION

1. Sign up

The sign up page will require user details to enter into the database. It will ask the user to provide details like the name, username, password, and mobile number to create a new user in the database. Regular expressions have been used to ensure that the details entered by the user are appropriate and in the right format. The sign up module will get the data from the user and then it will send the data to the database for storage.

2. Login and Authentication

The login and authentication module will allow the user to join in and chat from any device. The user will be asked to enter the username chosen at the time of signup and the password chosen at the same time. These details will be cross checked with the user details in the database. If the details have matched successfully then the user will be redirected to the main chatting page with all the contacts and previous chats.

3. Contact

3.1 New Contact

The website will allow the user to store a new contact in the chat window. Creating the new contact involves adding the name and phone number. These will then be saved in the user database with all the other contacts and the user can access it anytime to chat with the new contact.

3.2 Existing Contact

This part will show the list of all the contacts present in the user's database. It will be displayed as just the name on the user's chatting page but will have the phone number too which the user can click to see.

4. Chat

3.1 Chat History

This part of the chat module will display all the previous chats between the two users. The previous chats will be stored and can be displayed or viewed again.

3.2 Live Chat

The part of the chat module connects the two users and allows them to view and send chats simultaneously. It will show who has sent the chat and to whom. This will get updated immediately on both the sender and the receiver side instantly.

SAMPLE CODE

NewRegister.jsx

```
import React, { useState } from "react";
import Nav from "../Navigation/NavBar";
import { Link } from "react-router-dom";
import Validate from "./formValidation";
import components from "./components";
import "./reg.css";
import { Connection } from "../SocketConnection/Connection";
import { SHA1 } from "crypto-js";

const conn = new Connection();

class ElementList {
  constructor(name, link) {
    this.name = name;
    this.link = link;
  }
}

let User = {};

// List of elements to be displayed in the nav bar
const navBarElements = [
  new ElementList("Home", "/"),
  new ElementList("Login", "/Register"),
  new ElementList("ContactUs", "/ContactUs"),
];

function NewRegister() {
  console.clear();
  let user = {};

  const signIN = () => {
    var form = document.forms[0].elements;

    for (var i = 0; i < form.length - 1; i++) {
      user[form[i].name] = form[i].value;
      if (form[i].name === "Password") {
        user[form[i].name] = SHA1(form[i].value).toString();
      }
    }
    conn.emit("NewRegister", user);
    user = {};
    console.log(user);
  };

  const [name, changeName] = useState("");
  function handleChange(e) {
    e.preventDefault();
    changeName(Validate(e.target));
    console.clear();
    console.log(e.target);
  }
  return (
    <div style={style}>
      <Nav navElements={navBarElements} />
      <div style={style2}>
        <div className="box">
          <h1>Sign In</h1>
          <form name="signup">
            {components.Signin.map((Signin) => (
              <div className="component">
                <label>{Signin.name}</label>
                <input
                  type={Signin.type}
                  name={Signin.name}
                  placeholder={Signin.placeholder}
                  id={Signin.id}
                  onChange={handleChange}
                />
              </div>
            ))}
            <div className="bg">
              <button onClick={signIN}>
                <Link to="/Register">Sign In</Link>
              </button>
            </div>
            <p>
              {components.SigningIn.content}
              <Link to={components.SigningIn.a}>
                {components.SigningIn.link}
              </Link>
            </p>
          </form>
          <p style={style3}>{name}</p>
        </div>
      </div>
    </div>
  );
}

//Styling for the main component
const style = {
  display: "flex",
  flexDirection: "column",
  alignItems: "center",
  justifyContent: "top",
  height: "100vh",
};

const btnLocation = {
  padding: "7px",
  borderRadius: "10px",
  width: "auto",
  alignSelf: "flex-end",
  marginTop: "10px",
  marginRight: "10px",
};

const style2 = {
  display: "flex",
  flexDirection: "column",
  alignItems: "center",
  height: "100%",
  width: "100%",
};
```

```
const style3 = {
  FontFace: "Arial",
  fontSize: "larger",
  padding: "10px",
};

export default NewRegister;
```

formValidation.js

```
//Using regular Expression for validation
var tempStore = "";
function Validate(data) {
  var err = "";
  if (data.value !== undefined) {
    if (data.name === "Name") {
      if (data.value.length < 3) {
        err = "Name must be at least 3 characters long";
      }
      if (data.value.length > 15) {
        err = "Name must be less than 15 characters long";
      }
      if (!data.value.match(/^[a-zA-Z ]+$/)) {
        err = "Name must be alphabets only";
      }
    }
    if (data.name === "Username") {
      if (data.value.length < 3) {
        err = "Username must be at least 3 characters long";
      } else if (data.value.length > 15) {
        err = "Username must be less than 15 characters long";
      } else if (!data.value.match(/^[a-zA-Z0-9]+[._]*[a-zA-Z0-9]*$/)) {
        err =
          "Username must be alphanumeric only + Special Characters like '.'
and '_' allowed in middle or in the end";
      }
    }

    if (data.name === "Password") {
      if (data.value.length < 6) {
        err = "Password must be at least 6 characters long";
      } else if (data.value.length > 15) {
        err = "Password must be less than 15 characters long";
      } else if (
        !data.value.match(

/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{6,}$/
        )
      ) {
        err =
          "Password must be alphanumeric and must have atleast one special
Character";
      }
      tempStore = data.value;
    }

    if (data.name === "Confirm Password") {
      if (data.value !== tempStore) {
        err = "Password and Confirm Password must be same";
      }
    }

    if (data.name === "Email") {
      if (
        !data.value.match(

/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-
9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/
        )
      ) {
        err = "Email is invalid";
      }
    }
  }
  return err;
}

export default Validate;
```

reg.css

```css
@media (min-width: 768px) {
    .box {
      width: 350px;
      margin: 0 auto;
    }
  }
  .box {
    display: "flex";
    flex-direction: "column";
    align-self: center;
    width: fit-content;
  }
  input {
    padding: 5px;
    border-radius: 5px;
    border: none;
    margin: 1vh;
    outline: none;
  }
  a {
    text-decoration: none;
  }
  form button {
    padding: 5px;
    border-radius: 5px;
    border: none;
    margin: 1vh;
  }

  .component {
    display: flex;
    justify-content: space-between;
    margin: 0 2vw;
    gap: 3vw;
  }
  .box {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #a419d4;
    padding: 2vh;
    border-radius: 10px;
    margin-top: 10vh;
    justify-content: center;
  }
  .component {
    width: 100%;
    align-items: center;
    justify-content: space-between;
  }
  form {
    display: flex;
    flex-direction: column;
    align-items: center;
  }
  h1 {
    margin-bottom: 5vh;
  }
  .bg {
    margin-top: 5vh;
    display: flex;
    width: 100%;
    justify-content: space-around;
  }

  .bg button {
    font-size: larger;
    padding: 10px;
  }
```

package.json

```json
{
    "name": "server",
    "version": "0.0.0",
    "private": true,
    "scripts": {
      "start": "nodemon index.js"
    },
  "dependencies": {
    "bcrypt-nodejs": "^0.0.3",
    "cookie-parser": "~1.4.4",
    "cors": "^2.8.5",
    "crypto-js": "^4.1.1",
    "debug": "~2.6.9",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "express-handlebars": "^6.0.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "mongoose": "^6.0.13",
    "morgan": "~1.9.1",
    "mysql": "^2.18.1",
    "nodemon": "^2.0.15",
    "socket.io": "^4.3.0",
    "time": "^0.12.0"
  },
  "main": "index.js",
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Complete Code for Table Creation:

https://github.com/Internet-And-Web-Programming/ChatStrive-React/blob/master/server/Models/queries.sql

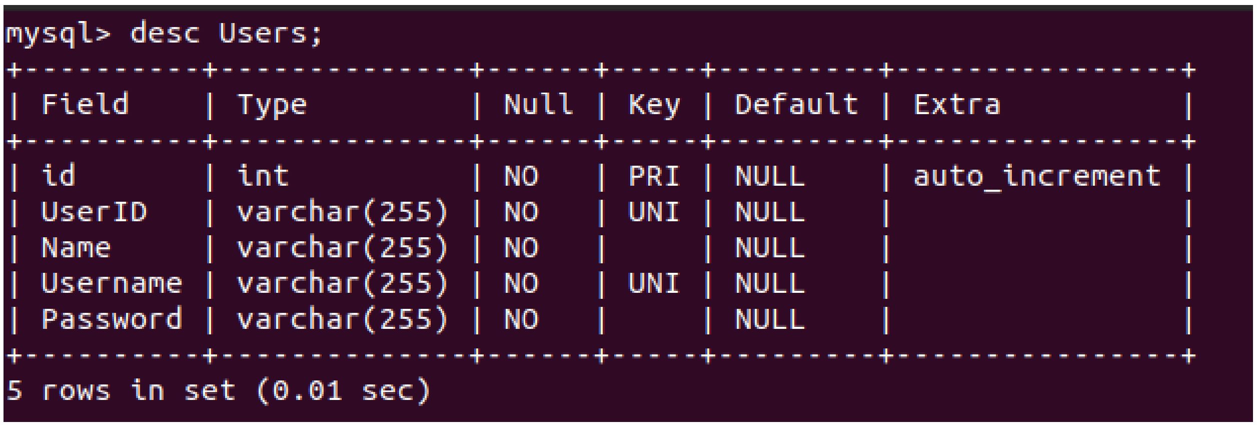
Complete Backend API Code:

https://github.com/Internet-And-Web-Programming/ChatStrive-React/blob/master/MySQL%20backend%20API/storageAPI.js

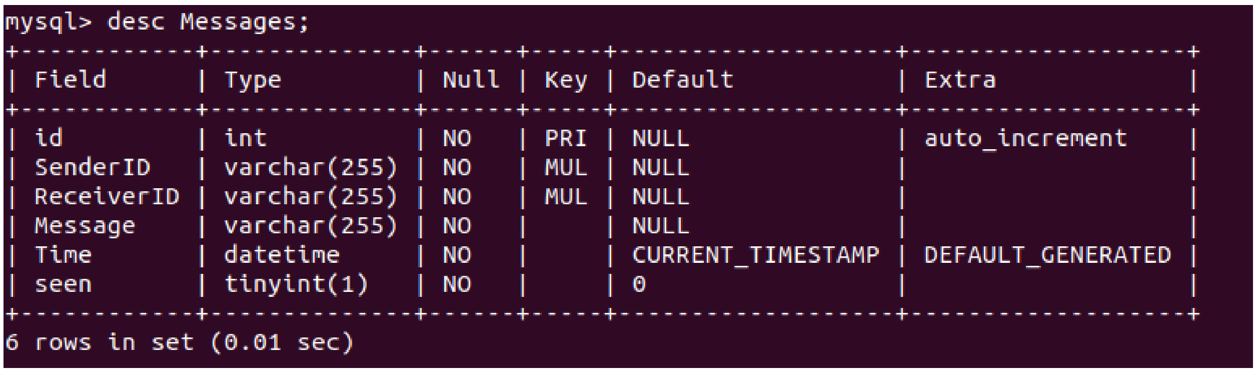
Complete Project Code:

https://github.com/Internet-And-Web-Programming/ChatStrive-React

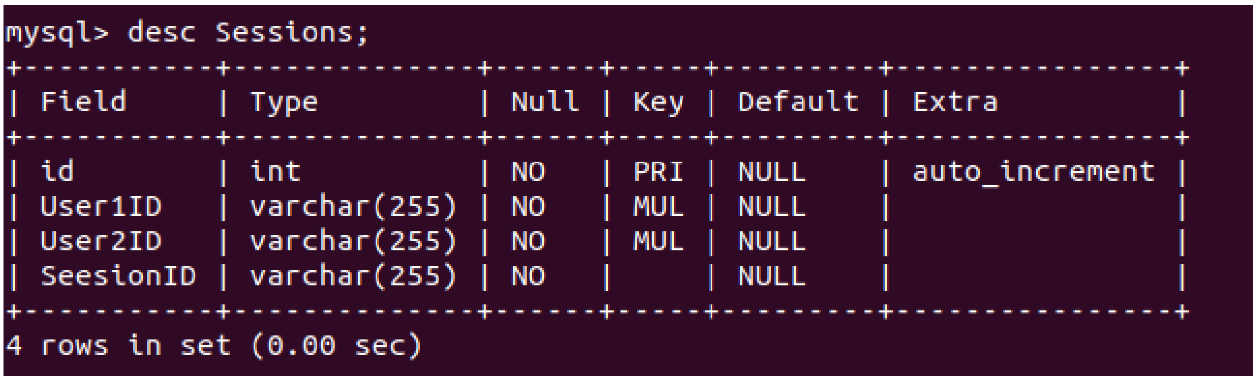
SCREENSHOTS



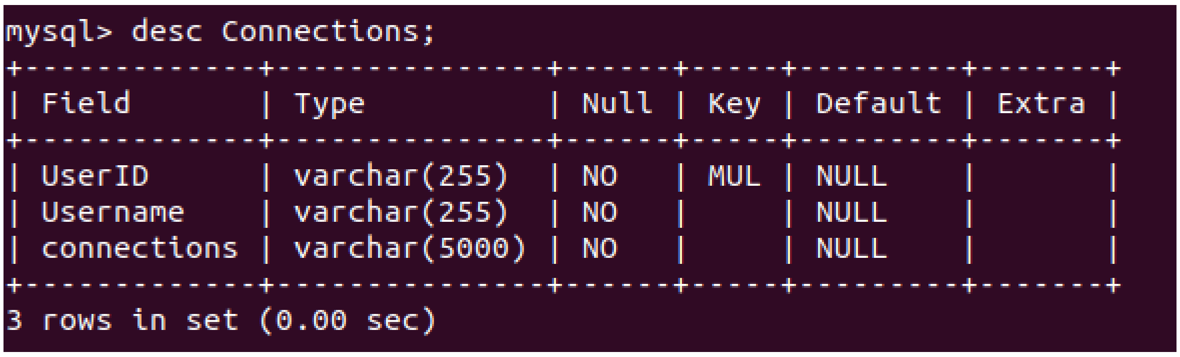
```
mysql> desc Users;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| UserID   | varchar(255) | NO   | UNI | NULL    |                |
| Name     | varchar(255) | NO   |     | NULL    |                |
| Username | varchar(255) | NO   | UNI | NULL    |                |
| Password | varchar(255) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```



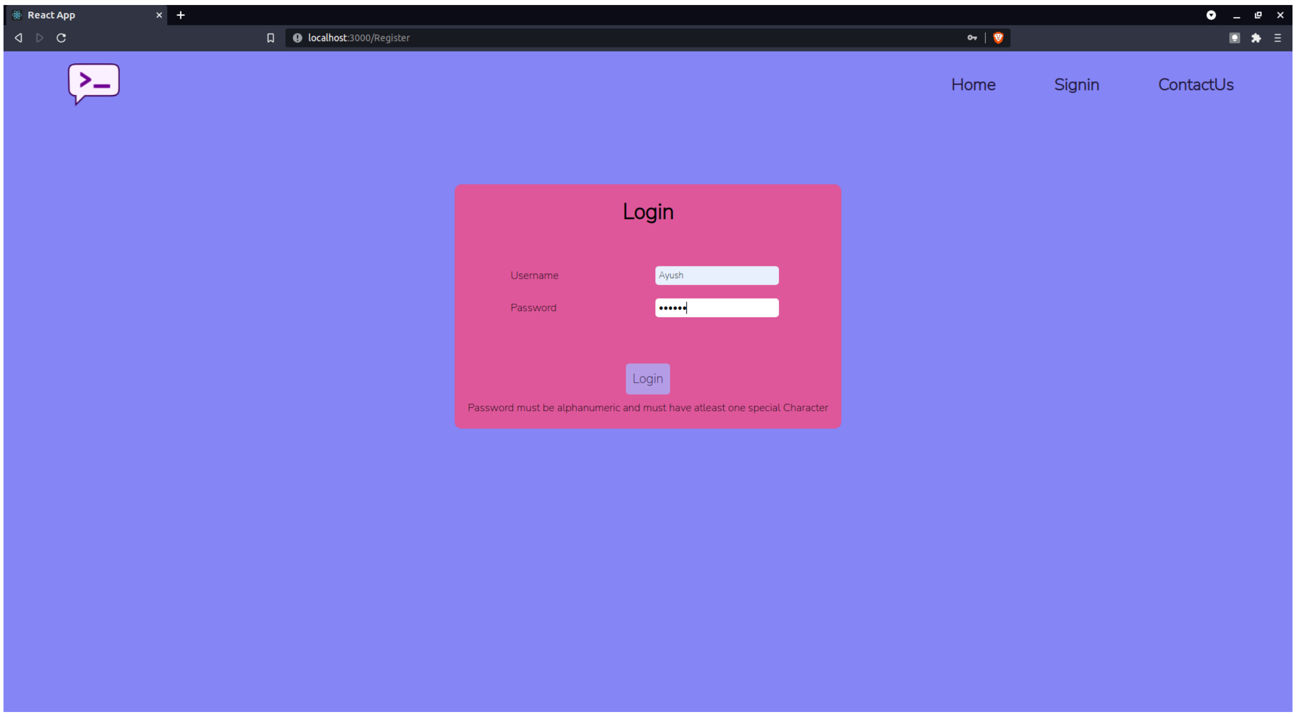
```
mysql> desc Messages;
+------------+--------------+------+-----+-------------------+-------------------+
| Field      | Type         | Null | Key | Default           | Extra             |
+------------+--------------+------+-----+-------------------+-------------------+
| id         | int          | NO   | PRI | NULL              | auto_increment    |
| SenderID   | varchar(255) | NO   | MUL | NULL              |                   |
| ReceiverID | varchar(255) | NO   | MUL | NULL              |                   |
| Message    | varchar(255) | NO   |     | NULL              |                   |
| Time       | datetime     | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| seen       | tinyint(1)   | NO   |     | 0                 |                   |
+------------+--------------+------+-----+-------------------+-------------------+
6 rows in set (0.01 sec)
```



```
mysql> desc Sessions;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int          | NO   | PRI | NULL    | auto_increment |
| User1ID   | varchar(255) | NO   | MUL | NULL    |                |
| User2ID   | varchar(255) | NO   | MUL | NULL    |                |
| SeesionID | varchar(255) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```



```
mysql> desc Connections;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| UserID      | varchar(255)  | NO   | MUL | NULL    |       |
| Username    | varchar(255)  | NO   |     | NULL    |       |
| connections | varchar(5000) | NO   |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```


React App

localhost:3000/Register

Home Signin ContactUs

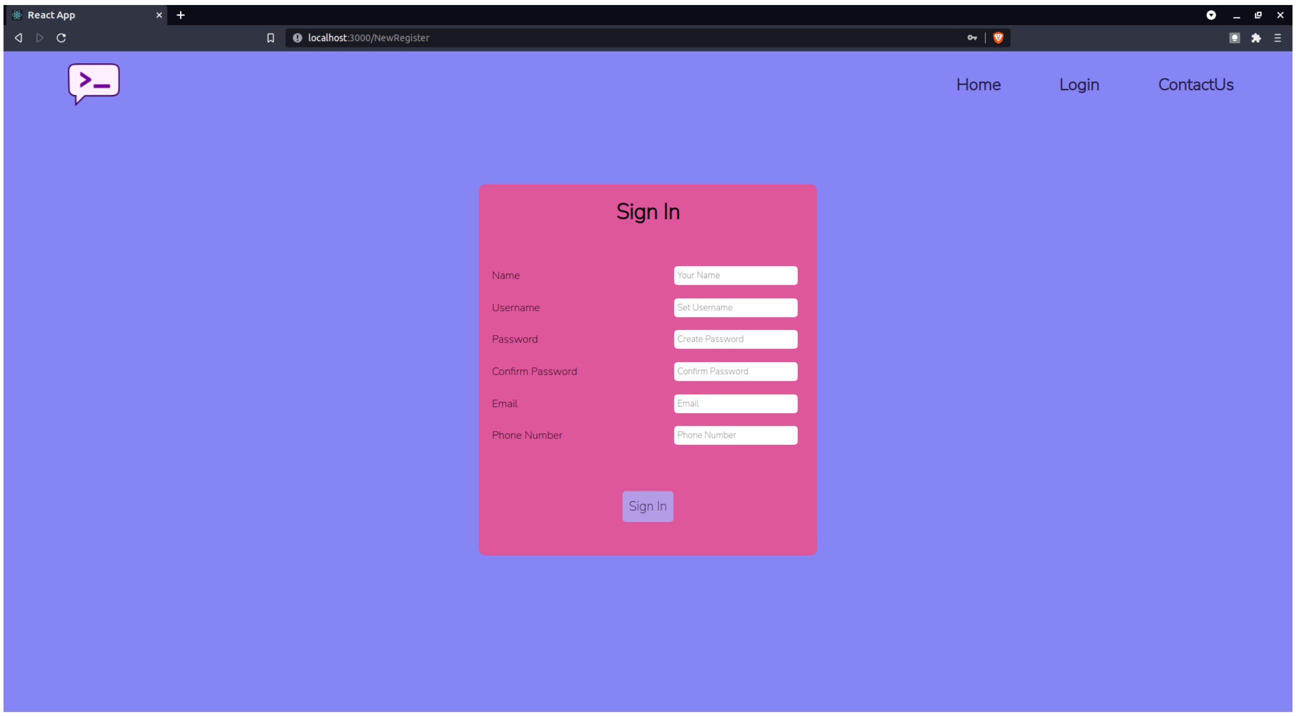
Login

Username Ayush

Password ••••••

Login

Password must be alphanumeric and must have atleast one special Character


React App

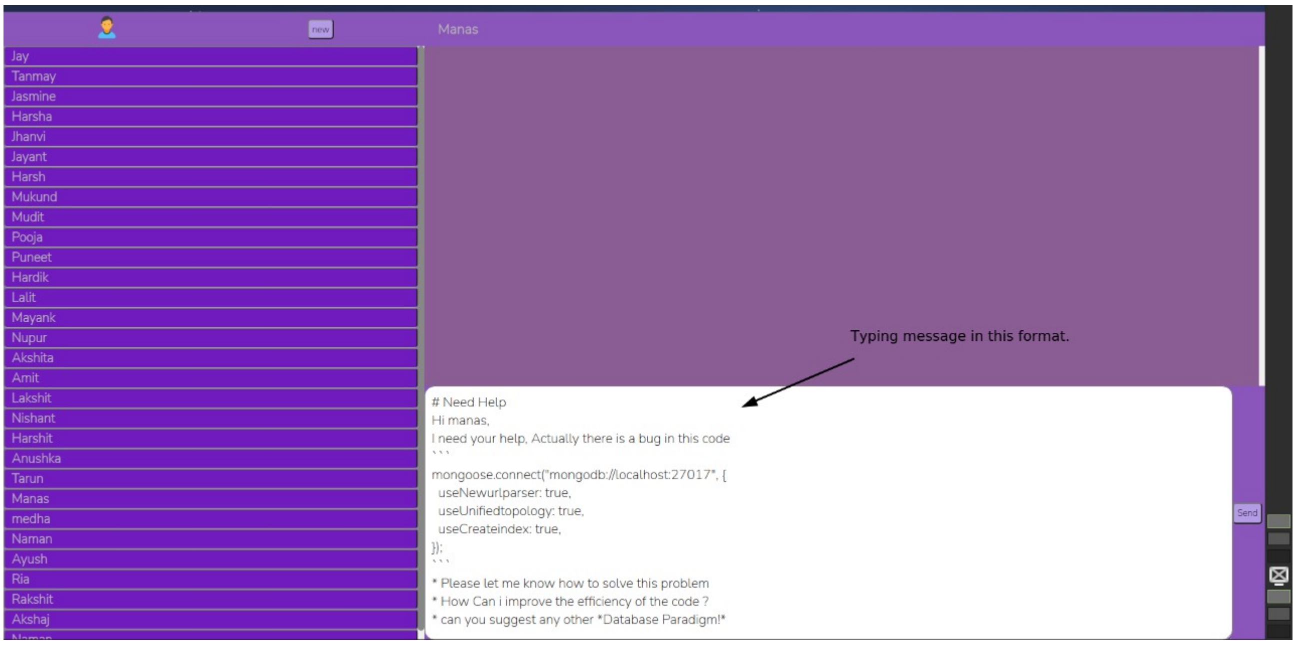
localhost:3000/NewRegister

Home Login ContactUs

Sign In

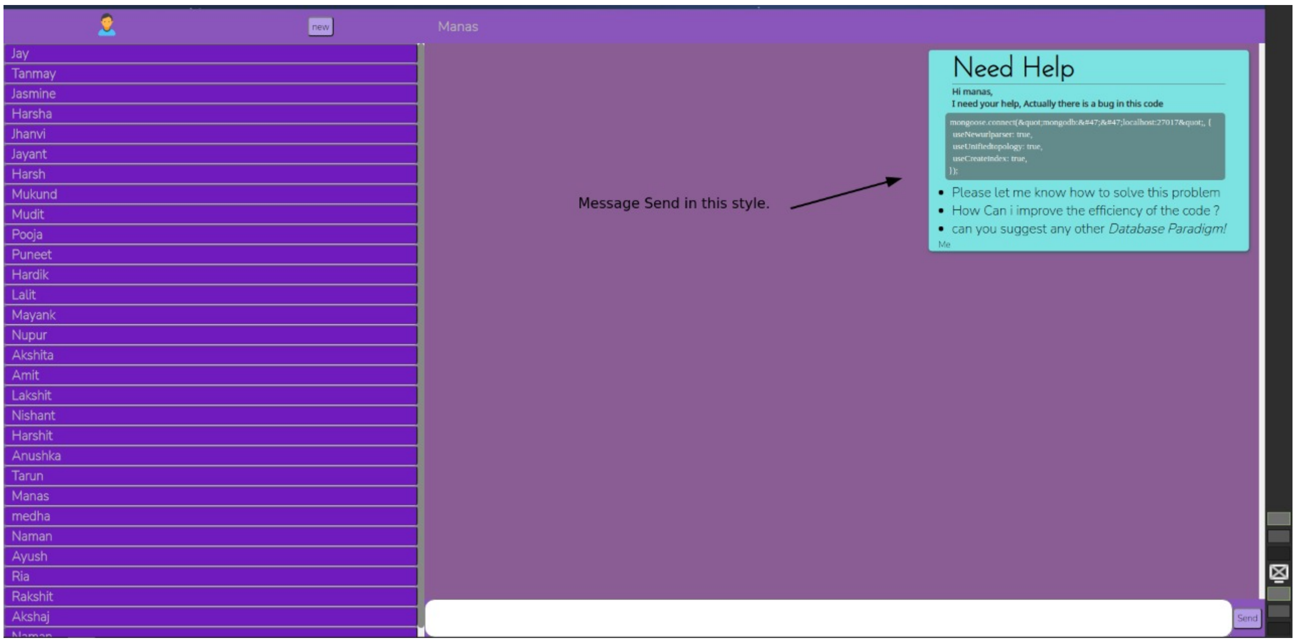
Name	Your Name
Username	Set Username
Password	Create Password
Confirm Password	Confirm Password
Email	Email
Phone Number	Phone Number

Sign In



Typing message in this format.

```
# Need Help
Hi manas,
I need your help, Actually there is a bug in this code
```
mongoose.connect("mongodb://localhost:27017", {
  useNewurlparser: true,
  useUnifiedtopology: true,
  useCreateindex: true,
});
```
* Please let me know how to solve this problem
* How Can i improve the efficiency of the code ?
* can you suggest any other *Database Paradigm!*
```



Message Send in this style.

Need Help

Hi manas,
I need your help, Actually there is a bug in this code

```
mongoose.connect(&quot;mongodb:&#47;&#47;localhost:27017&quot;, {
 useNewurlparser: true,
 useUnifiedtopology: true,
 useCreateindex: true,
});
```

- Please let me know how to solve this problem
- How Can i improve the efficiency of the code ?
- can you suggest any other *Database Paradigm!*

Me

CONCLUSION

The project successfully delivered on all requirements of an online messaging application. Care was ensured during the design to make sure data integrity is maintained and to avoid all forms of redundancies associated with data. The user is assured a very friendly interface and the project was made with the idea of keeping the users comfortable.

This project was built in such a manner that any modifications in the future can be implemented without affecting the pre-existing functionality of the application. The limitation of this project would be it requires internet access at all times. For future scope, we could include sending of animations and enhancing the text font.

Messaging apps have more global users than other traditional social networks which means they would play an increasingly vital role in the distribution of information in the future. As they are low cost and easily available, the number of users has grown tremendously.