BuddyUp – Real Time Chatting Website

A

Synopsis

Submitted in partial fulfilment of the requirement for the award of Degree of

BACHELOR OF COMPUTER APPLICATIONS

Submitted to

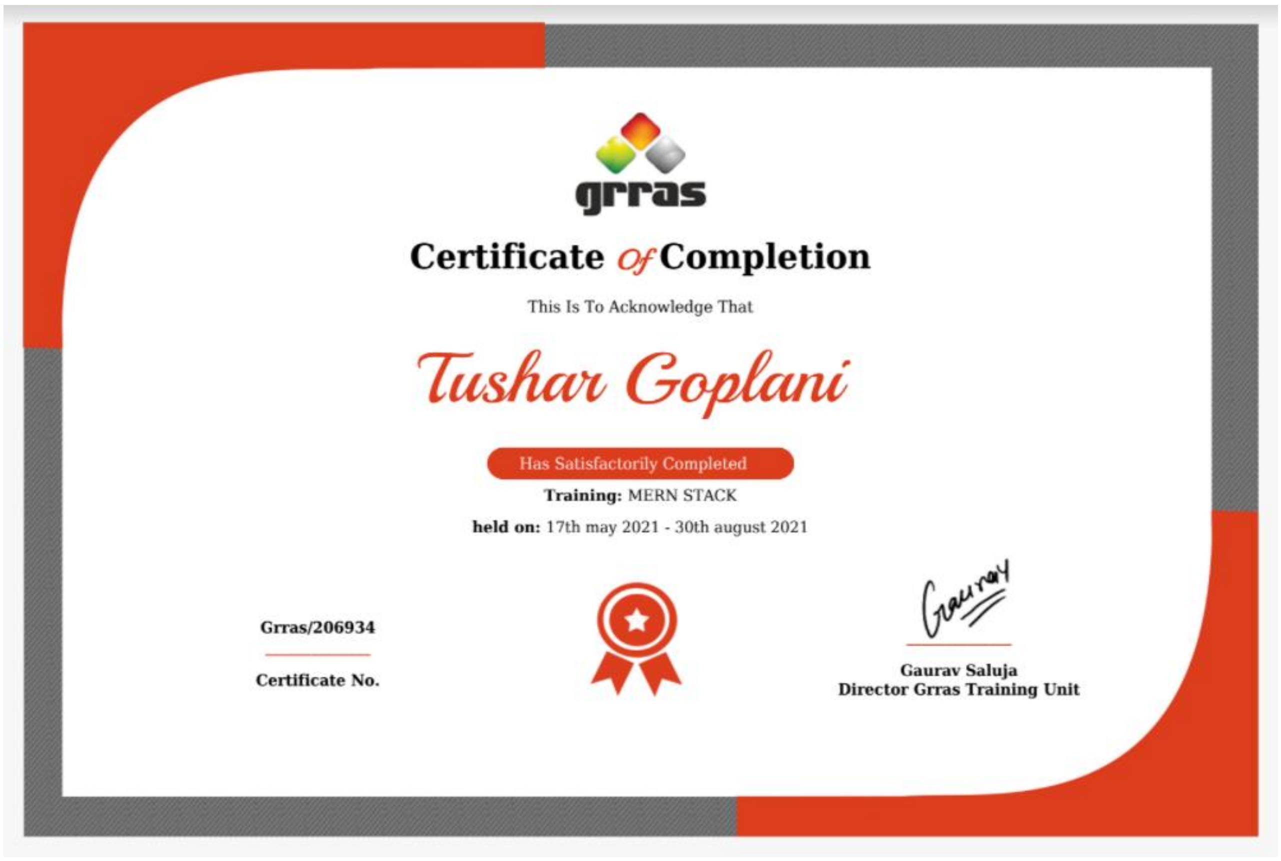
ALANIYA, KOTA (Raj.)

Submitted by

Tushar Goplani (K18420)
B.C.A. (5th semester)

SCHOOL OF COMPUTER APPLICATION
CAREER POINT UNIVERSITY, ALANIYA, KOTA

TRAINING CERTIFICATE



grras

Certificate Of Completion

This Is To Acknowledge That

Tushar Goplani

Has Satisfactorily Completed

Training: MERN STACK

held on: 17th may 2021 - 30th august 2021

Grras/206934

Certificate No.

[illegible]

Gaurav Saluja
Director Grras Training Unit

CERTIFICATE

I am **Tushar Goplani S/O Mr. Hemant Goplani** of B.C.A Vth Semester herby declare that the work, which is being presented in the Internship Project Report, entitled **"BuddyUp: Real Time Chatting Website"** in partial fulfilment for the award of Degree in "Bachelor of Computer Applications" and submitted to the Department of Computer Applications of Career Point University, Kota is a record of my work carried under the guidance of **"Mr. Sanjay Rathore (Grras Solutions Pvt Ltd, Jaipur)".**

Mr. Sanjay Rathore
MERN Stack Trainer
Grras Solutions, Jaipur

ACKNOWLEDGEMENT

It’s my esteemed pleasure to present this internship project report on **Real Time Chatting Website**. I would like to thanks Mr Sanjay Rathore (Web Developer of Grras Solutions) for providing us all necessary facilities to carry the project successfully.

I would like to thanks all the faculty members of Computer Application Department for helping me to successfully develop this project.

Last but no least we are thankful to my friends without whose support at various stages, this project wouldn’t have materialized. I thank to all the supporting friends who directly or indirectly helped me in completing this project.

DECLARATION BY CANDIDATE

I **Tushar Goplani** student of **Computer Application Department, CAREER POINT UNIVERSITY, KOTA** herby Declare that the work presented in this project is outcome of my own work, is bonfire, correct to the best of my knowledge and this work has been carried out taking care of IT Ethics. This work presented does not infringe and patented work has been submitted to any university for the award of any degree.

Tushar Goplani
KID: k18420
B.C.A. (Vth Sem)

TABLE OF CONTENTS

INTRODUCTION

This website is implemented in MERN Stack. This project is targeted for two-way conversations between users. It is made up of two applications the client application, which runs on the user's pc and server application, which runs on any pc on the network. User can add friends and remove friends. And user can delete messages or can clear chat history. And can see message information like date and time. And user also edit their profile picture and can edit their personal information like name, email and change password.

OBJECTIVE

- The basic concept of the application is to allow the users to chat virtually using the Internet.
- We can send messages easily to any person at any time.
- It is fast because of real time chat.
- Users can easily use this site because of its easy user interface.

SCOPE

- This chatting website is used for communication through internet for users.
- User can login their account and they start chatting to their friends by sending friend request.
- Users can update their details.
- This site will run the entire work in a much better, accurate and error free manner.

FEASIBILITY STUDY

The feasibility study plays a major role in the analysis of the system. The very decision of the system analysis whether he should design a particular system or not and till what extend or limits the very project should be stretched can be inferred from The feasibility system can be categorized into the system feasibility study. Hence the feasibility study forms the basis of the system.

1) TECHNICAL FEASIBILITY

It determines the technology needed for the proposed system is available and how this technology can be integrated into the organization. Genuine Computer is equipped with the necessary hardware and software.

2) Organizational and Culture Feasibility

The organizational and culture feasibility analysis is done to scrutinize whether the system sits up with the working environment and the organizational discipline and rules.

Following issues are considered:
• The current level of computer competency.
• Re-engineering the old working procedures.
• Substantial tech phobia
• Expectation of the users.

3) Operational Feasibility

The proposed system will automated and make it user friendly. With the required training the users will find the system easier to operate. The systems cuts down the time delay.

4) Economic Feasibility

The economic feasibility of the system looks upon the financial aspects of the system. It determines whether the project is economically feasible. In other words. It determines whether the investment that goes into the implementation of the project is recoverable. The cost benefit analysis is a commonly used method in evaluating the effectiveness of the system. As the hardware is already available and no investment is to be made in that direction, the only cost involved is that of implementing the system and software.

SYSTEM DESIGN

In this website, first of all user have to signup their account by entering name, username, password and mail id. After entering these details user get OTP on mail. And if user enter correct OTP then user’s account will be created. And he/she will redirect to login page. After login user can create friends by sending friend request. After accepting their request users can chat. And user can see message’s date and time. And can delete message or can clear chat history. And user can unfriend their friends. And in the setting page user can set their profile picture and can change password or edit name or mail id.

TECHNICAL SPECIFICATION

At Developer Side

- **Software Used:**

  - Operating system : Windows 10
  - Fronted - HTML ,CSS, Bootstrap, React, Redux, JSX and socket.
  - Backend – NodeJS , Express, MongoDB (atlas).
  - Tools Used : VS Code, Node and Browser
  - Hosting Service : AWS S3 bucket

- **Hardware Used:**

  - Processor : Intel Core i3 7th gen
  - RAM : 8 GB
  - Hard Disk : 1 TB

Significance features of Language Used

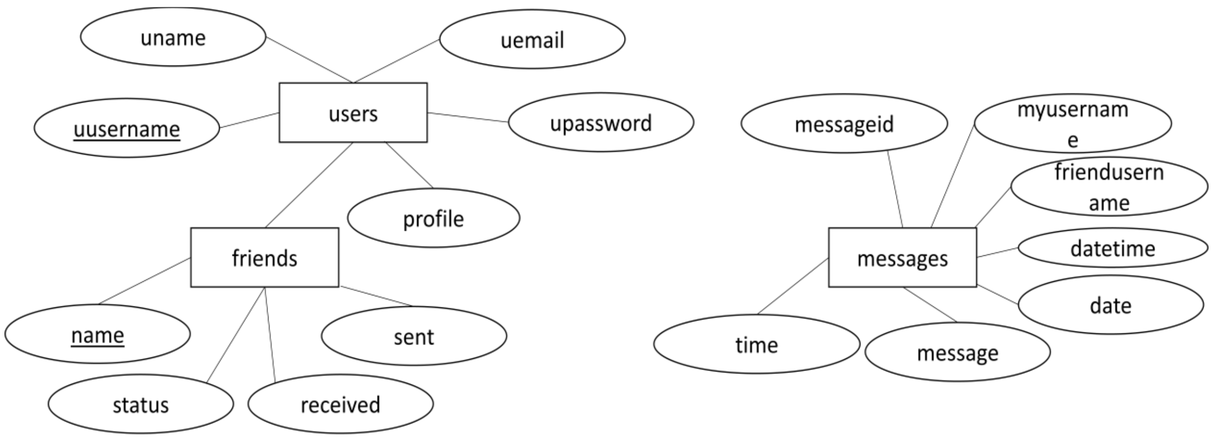
JavaScript is a client side technology, it is mainly used for gives client side validation, but it have lot of features which are given below -

- JavaScript is a object-based scripting language.
- Giving the user more control over the browser.
- It Detecting the user's browser and OS,
- It is light weighted.
- JavaScript is a scripting language and it is not java.
- JavaScript is interpreter based scripting language.
- JavaScript is case sensitive.
- JavaScript is object based language as it provides predefined objects.
- Every statement in javascript must be terminated with semicolon (;).
- Most of the javascript control statements syntax is same as syntax of control statements in C language.
- An important part of JavaScript is the ability to create new functions within scripts. Declare a function in JavaScript using function keyword.
- Working in the MERN stack delivers powerful results simply and efficiently. Knowing your way around the stack is an important skill set since building and deploying solid MERN apps is likely to gain importance in the future.
- Node.js was built on Chrome’s JavaScript runtime to make it more conducive to building fast-operating network applications with easy scalability. The platform operates using a nonblocking, event-driven I/O model that is incredibly efficient.
- Node.js’s efficiency and simplicity make it an ideal platform for real-time applications running across distributed devices, especially those with intense data requirements. But Node.js has far more potential in conjunction with the MERN stack.

Description of Modules

1. **Sign up**: Users can create their new account by enter their details.
2. **Sign in**: Users can login their account by enter their username and password.
3. **Forgot Password**: Users can forgot their password by enter their email id.
4. **Add Friend**: Users can send friend request by enter friend’s username.
5. **Unfriend**: Users can remove their friend.
6. **Notification**: User will receive friend request by notification.
7. **Accept / Decline Request**: User can accept or decline request.
8. **Update account details**: User can update their account details like their profile pic, name, password and email.
9. **Chat**: Users can chat with their friends.
10. **Message Info**: User can see date and time of messages.
11. **Delete Message**: User can delete messages from chats.
12. **Clear Chat History**: User can delete all the messages of chats by one single click.

ER Diagram



Note: users and messages are two different collections of MongoDB.

DATABASE

1. Collection name – users

Field name	Datatype	Description
uusername	Text	User’s Username
upassword	Text	User’s Password
uname	Text	User’s Name
uemail	Text	User’s Email
profile	Text	User’s Profile
friends	Array	User’s Friend

2. Collection name – messages

Field name	Datatype	Description
myusername	Text	Sender username
friendusername	Text	Receiver username
message	Text	Message
messageid	Number	Message Id
date	Text	Message date
time	Text	Message time
datetime	Text	Message date and time

Code

Complete source code on GitHub: github.com/tushargoplani/buddyup1

Frontend Code

Index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="logo1.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
    <!-- <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script> -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <!-- Bootstrap core CSS -->
    <link href="dist/css/lib/bootstrap.min.css" type="text/css" rel="stylesheet">
    <!-- Swipe core CSS -->
    <link href="dist/css/swipe.min.css" type="text/css" rel="stylesheet">
    <!-- Favicon -->
    <link href="dist/img/favicon.png" type="image/png" rel="icon">
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>BuddyUp - Make Friends</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <script src="dist/js/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script>window.jQuery || document.write('<script
src="dist/js/vendor/jquery-slim.min.js"><\/script>')</script>
    <script src="dist/js/vendor/popper.min.js"></script>
    <script src="dist/js/swipe.min.js"></script>
    <script src="dist/js/bootstrap.min.js"></script>
  </body>
</html>
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { Provider } from 'react-redux'
import store from '../src/Store/store';



ReactDOM.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Store.js

```
import { createStore, applyMiddleware,combineReducers } from 'redux';
import { composeWithDevTools } from 'redux-devtools-extension';
import thunk from 'redux-thunk';
import {user} from '../reducers/userReducer';
import {loading} from '../reducers/loading';
```

```
var rootReducer = combineReducers({user,loading});
var store = createStore(rootReducer,
    composeWithDevTools(applyMiddleware(thunk))
    );
export default store;
```

userreducer.js

```
export function user (state=null,action){
    switch(action.type)
    {
        case "LOGIN_USER":
                return action.payload;

        case "LOGOUT_USER":
           return null;
        default:
           return state;
    }
}
```

Loading.js

```
export function loading (state=false,action){
    switch(action.type)
    {
        case "LOADING_TRUE":
                return true;

        case "LOADING_FALSE":
           return false;
        default:
           return state;
    }
}
```

Nav.js

```
import React from 'react';
import '../App.css';
import {BrowserRouter as Router,Switch,Route,NavLink} from 'react-router-dom';
import './css/nav.css';
import Login from './login';
import Signup from './signup';
import Chatpage from './chatpage';
import {useDispatch, useSelector} from 'react-redux';

function Nav(props) {

    const user = useSelector(state => state.user);
    const dispatch = useDispatch();
    function logout(){
            dispatch({type:"LOGOUT_USER"});
            // props.history.push("/");
    }

    return (
        <Router>
        <div>

            <nav class="navbar navbar-expand-md  navbar-dark"
style={{backgroundColor: "black"}}>
            <a class="navbar-brand" href="#"><div
id="logo">Buddy<span>Up</span></div></a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#collapsibleNavbar">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="collapsibleNavbar">
                <ul class="navbar-nav ml-auto">
                {user && <li class="nav-item">
                    <li class="nav-link text-white font-weight-bold"><NavLink
class="text-white font-weight-bold" onClick={logout} exact
to="/">Logout</NavLink></li>
                </li>}
                </ul>
            </div>
            </nav>

            <div>
                <Switch>
                    <Route path="/" exact component = {Login}/>
                    <Route path="/create-account" exact component = {Signup}/>
                    <Route path="/buddyup-chat" exact component = {Chatpage}/>
                </Switch>
                </div>

        </div>
        </Router>
    )
}
export default Nav
```

login.js

```
import React,{useState,useEffect} from 'react';
import './css/login.css';
import axios from 'axios';
import {Switch,Route,NavLink} from 'react-router-dom';
import Signup from './signup';
import Chatpage from './chatpage'
import { checkLogin } from '../actions/userAction';
import { useDispatch, useSelector} from 'react-redux';

function Login(props) {

  const [users, setUser] = useState([]);
  const [password, setpassword] = useState("");
  const [username, setusername] = useState("");
  const [email, setemail] = useState("");

  const reduxUser = useSelector(state => state.user);

  useEffect(() => {
    if(reduxUser){
      props.history.push("/buddyup-chat");

    }

  }, [reduxUser]);


  useEffect(()=>{
   axios.get('http://localhost:3000/list-account').then((res)=>{
      //  console.log(res.data.data)
       setUser(res.data.data)
   })
  },[])

  const dispatch = useDispatch();

  function setValue(e) {
    e.target.name=="username" && setusername(e.target.value);
    e.target.name=="password" && setpassword(e.target.value);
    e.target.name=="email" && setemail(e.target.value);
  }
  function Sendpassword(){
    alert(email)
   var  s={email}
    axios.post('http://localhost:3000/user-by-email',s).then((res)=>{
        console.log(res.data.data)
       alert("completed")
})
}

  function Auth(){
      // alert(username);
      // alert(password);
      dispatch(checkLogin({username,password}));
  }

function goToSignup(){
    var Login = document.getElementById('mainlogin');
    Login.style.display = "none";
}



function goToForgot(){
  let Login = document.getElementById('login');
  Login.style.display = "none";
  let forgot = document.getElementById('forgotcredentials');
  forgot.style.display = "block";
}

function forgottologin(){
  let Login = document.getElementById('login');
  Login.style.display = "block";
  let forgot = document.getElementById('forgotcredentials');
  forgot.style.display = "none";
}

  return (
  <React.Fragment>
   <div id="mainlogin">
  <div class="left leftflex">

    <div id="login">
      <h2>Sign in your account</h2>
      <form action="">
        <i class="fa fa-user"></i>
        <input name="username" value={username} onChange={(e)=>{setValue(e);}}
type="text" placeholder="Enter your username" class="lowercase"/><br/> <br/>
<br/>
```

```
        <i class="fa fa-key"></i>
        <input name="password" value={password} onChange={(e)=>{setValue(e);}}
type="password" placeholder="Enter your password"/> <br/><br/>
        <span class="text-primary" onClick={goToForgot}
style={{cursor:'pointer'}}> Forgot Password ?</span><br/><br/>
           <button type="button" onClick={Auth}>Sign in  </button>
         </form>
         <div id="createact">Don't have account? <NavLink exact to="/create-
account" onClick={goToSignup} class="text-primary">Create
Account</NavLink></div>
       </div>

       <div id="forgotcredentials">
         <h2>Forgot Credentials</h2>
         <form action="">
           <i class="fa fa-envelope"></i>
           <input name="email" value={email} onChange={(e)=>{setValue(e);}}
type="email" placeholder="Enter your email"/> <br/><br/>
           <a class="text-primary" href="" onClick={forgottologin}> Back to
Signin </a><br/><br/>
           <button onClick={Sendpassword} type="button" > Send me </button>
         </form>
       </div>
     </div>

     <div class="right">
      <div id="signup">
         <h1>Hello Dear!</h1> <br/> <br/>
         <p>Enter your personal details and start your journey with
<b>BuddyUp</b> today.</p> <br/> <br/>
         <button ><NavLink exact to="/create-account"
onClick={goToSignup}>Sign up</NavLink> </button>
       </div>
     </div>
   </div>

 <div>
 <Switch>
      <Route path="/create-account" exact component = {Signup}/>
      <Route path="/buddyup-chat" exact component = {Chatpage}/>
 </Switch>
 </div>

 </React.Fragment>
    )
}
 export default Login
```

signup.js

```
import React, { useState } from 'react';
import './css/signup.css';
import { BrowserRouter as Router, Switch, Route, NavLink } from 'react-router-
dom';
import Login from './login';
import axios from 'axios';

function Signup(props) {

  function goToSignin() {
    var Signup = document.getElementById('mainsignup');
    Signup.style.display = "none";
  }

  const [uname, setuname] = useState("");
  const [uemail, setuemail] = useState("");
  const [upassword, setupassword] = useState("");
  const [uusername, setuusername] = useState("");
  const [otp, setotp] = useState("");
  const [randomotp, setrandomotp] = useState("")

  function setValue(e) {
    e.target.name === "Uname" && setuname(e.target.value);
    e.target.name === "Uemail" && setuemail(e.target.value);
    e.target.name === "Upassword" && setupassword(e.target.value);
    e.target.name === "Uusername" && setuusername(e.target.value);
    e.target.name === "otp" && setotp(e.target.value);
  }

  function validate() {
    var isvalid = true;

    // validate for Name
    if (uname == "" || uname == null) {
      isvalid = false;
      alert("please enter name");
    }
    //validate for email
    var emailregex =
/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/;
    if (!emailregex.test(uemail)) {
      alert("Email is not valid");
      isvalid = false;
    }
    //validate for username
    if (uusername == "" || uusername == null) {
      isvalid = false;
      alert("please enter username");
    }
    var userregex = /^[a-z0-9_\.]+$/;
    if (!userregex.test(uusername)) {
      alert("Usernames can only have: Lowercase Letters (a-z), Numbers (0-9),
Dots (.), Underscores (_)");
      isvalid = false;
    }
    //validate for password
    var passregex = /(?=(.*[0-9]))(?=.*[\!@#$%^&*()\\[\]{}\-
_+=~`|:;"'<>,./?])(?=.*[a-z])(?=(.*[A-Z]))(?=(.*)).{8,}/;
    if (!passregex.test(upassword)) {
      alert("Password should have 1 lowercase letter, 1 uppercase letter, 1
number, 1 special character and be at least 8 characters long");
      isvalid = false;
    }

    if (isvalid == true) {
      axios.post('http://localhost:3000/valid-username', { uusername
}).then((res) => {
        if (res.data.status == 'ok') {
          axios.post('http://localhost:3000/valid-email', { uemail
}).then((res) => {
            if (res.data.status == "ok") {
              var random = Math.floor((Math.random() * 1000000) + 1);
              setrandomotp(random);
              axios.post("http://localhost:3000/send-user-otp", { uemail, otp:
random }).then((res) => {
                if (res.data.status == "ok") {
                  alert("OTP sent to your mail id. Check your mail")
                  document.getElementById('login').style.display = "none";
                  document.getElementById('createotp').style.display =
"block";
                }
              })
            }
            else{
              alert("Account already created with this email id")
            }
          })
        }
        else{
          alert("Username already taken :( ");
        }
      })
    }

  }

  function otpcheck() {
    if(otp==randomotp){
    var s = { uname, uemail, uusername, upassword };
    console.log(s);
    axios.post('http://localhost:3000/create-account', s).then((res) => {
      alert(res.data.data);
      props.history.push("/");
    })
  }
  else{
    alert("Incorrect otp ");
    setotp('');
  }
  }

  function goback() {
    document.getElementById('login').style.display = "block";
    document.getElementById('createotp').style.display = "none";
  }


  return (
    // <Router>
    <React.Fragment>

      <div id="mainsignup">
        <div class="left leftflex">

          <div id="login">
            <h2>Create your account</h2>
            <form action="">
              <i class="fa fa-vcard"></i>
              <input name="Uname" value={uname} onChange={(e) => {
setValue(e); }} type="text" placeholder="Name" /> <br /><br />
              <i class="fa fa-envelope"></i>
              <input name="Uemail" value={uemail} onChange={(e) => {
setValue(e); }} type="email" placeholder="Email" /> <br /><br />
              <i class="fa fa-user"></i>
              <input name="Uusername" value={uusername} onChange={(e) => {
setValue(e); }} type="text" placeholder="Username" class="lowercase" /><br />
<br />
              <i class="fa fa-key"></i>
              <input name="Upassword" value={upassword} onChange={(e) => {
setValue(e); }} type="password" placeholder="Set Password" /> <br /><br />
```

```
              {/* <button onClick={sendData}> <NavLink exact to="/"
onClick={redirect}>  Sign up  </NavLink>  </button> */}
              <button type="button" onClick={validate}>  Sign up </button>
            </form>   <br /> <br />
            <div id="createact">Have an account? <NavLink exact to="/"
onClick={goToSignin} class="text-primary">Sign in</NavLink></div>
          </div>

          <div id="createotp">
            <h2>Enter OTP for Verfication</h2>
            <form action="">
              <i class="fa fa-vcard"></i>
              <input name="otp" value={otp} onChange={(e) => { setValue(e); }}
type="text" placeholder="Enter OTP " /> <br /><br />
              <button className='w-25 mr-4' type="button" onClick={goback}>
 Edit details </button>
              <button className='w-25 ml-4' type="button" onClick={otpcheck}>
 Validate </button>
            </form>
          </div>

        </div>

        <div class="right">
          <div id="signup">
            <h1>Welcome back!</h1> <br /> <br />
            <p>To keep connected with your friends login your <b>BuddyUp</b>
account here.</p> <br /> <br />
            <button ><NavLink exact to="/" onClick={goToSignin}>Sign
in</NavLink> </button>
          </div>
        </div>
      </div>

      <div>
        <Switch>
          <Route path="/" exact component={Login} />
        </Switch>
      </div>

    </React.Fragment>
    // </Router>
  )
}

export default Signup
```

chatpage.js

```
import React, { useEffect, useRef, useState } from 'react';
import axios from 'axios';
import { useSelector } from 'react-redux';
import Picker from 'emoji-picker-react';
import './css/chatpage.css';

function Chatpage(props) {

  const user = useSelector(state => state.user);
  useEffect(() => {
    if (!user) {
      props.history.push("/");
    }
  })

  const userId = useSelector(state => state.user._id);
  const userName = useSelector(state => state.user.uname);
  const userUserName = useSelector(state => state.user.uusername);
  const userPassword = useSelector(state => state.user.upassword);
  const userEmail = useSelector(state => state.user.uemail);
  var userImage = useSelector(state => state.user.profile);

  const [uname, setuname] = useState(userName);
  const [uemail, setuemail] = useState(userEmail);
  const [oldpassword, setoldpassword] = useState("");
  const [newpassword, setnewpassword] = useState("");
  const [reenternewpassword, setreenternewpassword] = useState("");
  const [uusername, setuusername] = useState(userUserName);
  const [friend, setfriend] = useState("");
  // const [profile, setprofile] = useState("");
  var profile;

  const [uploadPercentage, setuploadPercentage] = useState("");
  const [message, setmessage] = useState("");

  function setValue(e) {
    e.target.name === "Uname" && setuname(e.target.value);
    e.target.name === "Uemail" && setuemail(e.target.value);
    e.target.name === "oldpassword" && setoldpassword(e.target.value);
    e.target.name === "newpassword" && setnewpassword(e.target.value);
    e.target.name === "re-enter-new-password" &&
setreenternewpassword(e.target.value);
    e.target.name === "Uusername" && setuusername(e.target.value);
    e.target.name === "friend" && setfriend(e.target.value);
    e.target.name === "message" && setmessage(e.target.value);
  }
  function setProfile(e) {
    profile = e.target.files[0];
    console.log(profile);
  }

  function updateProfile() {
    if (profile != undefined) {
      var formData = new FormData();
      formData.append("_id", userId);
      formData.append("profile", profile);
      console.log(profile);
      axios.post('http://localhost:3000/update-profile', formData, {
        headers: {
          'Content-Type': 'multipart/form-data'
        },
        onUploadProgress: function (progressEvent) {
          console.log("file Uploading Progresss.......");
          // console.log(progressEvent);
          setuploadPercentage(parseInt(Math.round((progressEvent.loaded /
progressEvent.total) * 100)));
          //   setfileInProgress(progressEvent.fileName)
        }
      }).then((res) => {
        alert(res.data.data);
        userImage = profile;
      }).catch(res => {
        alert("Some issue occur while updating your profile");
      });
    }
    else {
      alert("please choose profile");
    }
  }

  function updateDetails() {
    var updateUserDetails = { userId, uname, uemail };
    axios.post('http://localhost:3000/update-user',
updateUserDetails).then((res) => {
      alert(res.data.data);
    }).catch(res => {
      alert("Edit not saved :(");
    })
  }

  function changePassword() {
    if (userPassword == oldpassword) {
      var isvalid = true;
      //validate for password
      var passregex = /(?=(.*[0-9]))(?=.*[\!@#$%^&*()\\[\]{}\-
_+=~`|:;"'<>,./?])(?=.*[a-z])(?=(.*[A-Z]))(?=(.*)).{8,}/;
      if (!passregex.test(newpassword)) {
        alert("Password should have 1 lowercase letter, 1 uppercase letter, 1
number, 1 special character and be at least 8 characters long");
        isvalid = false;
      }
      if (isvalid == true) {
        if (newpassword == reenternewpassword) {
          axios.post('http://localhost:3000/update-password', { userId,
reenternewpassword }).then((res) => {
            alert(res.data.data);
            setoldpassword('');
            setnewpassword('');
            setreenternewpassword('');
          }).catch(res => {
            alert("Password not changed :( ");
          })
        }
        else {
          alert("New password not match with re-enter new password")
        }
      }
    }
    else {
      alert("Old password is incorrect")
    }
  }

  function addFriend() {
    if (friend != "") {
      if (friend != userUserName) {
        axios.post('http://localhost:3000/search-for-user', { friend }).then(
          (res) => {
            if (res.data.data.length > 0) {
              axios.post('http://localhost:3000/get-notif', myUsername).then(
                (res) => {
                  if (res.data.data[0].friends) {
                    if (res.data.data[0].friends.length >= 1) {
                      var forUnique = res.data.data[0].friends.map((a) => {
return a.name });
                      var status = forUnique.some(elem => elem === friend);
                      if (status == true) {
                        alert("you cant send friend request again")
                      }
                      else {
                        // id validation that user exist or not
                        var addfrnd = { friend, userUserName };
                        axios.post('http://localhost:3000/add-friend',
addfrnd).then((res) => {
                          alert(res.data.data);
                          setfriend('');
                        })
                      }
                    }
                  }
                  else {
                    var addfrnd = { friend, userUserName };
                    axios.post('http://localhost:3000/add-friend',
addfrnd).then((res) => {
                      alert(res.data.data);
                      setfriend('');
                    })

                  }
                })
            }
            else {
              alert("User not found. Please enter correct username.")
            }
          })
      }
      else {
        alert("You can't send you a request. 🥱")
      }
    }
    else {
      alert("enter frnd's username");
    }
  }

  function accept(friendReq) {
    // console.log(friendReq);
    var acceptFrnd = { friendReq, userUserName };
    // console.log(acceptFrnd);
    axios.post('http://localhost:3000/accept-request', acceptFrnd).then((res)
=> {
      if (res.data.status == "ok") {
        alert(res.data.data);
      }
    });
  }

  function decline(mine, frnd) {
    axios.post('http://localhost:3000/unfriend-or-decline', { mine, frnd
}).then((res) => {
      if (res.data.status == "ok") {
        alert(res.data.data);
      }
    });
  }

  function unfriend(mine, frnd) {
    axios.post('http://localhost:3000/unfriend-or-decline', { mine, frnd
}).then((res) => {
      if (res.data.status == "ok") {
        alert(res.data.data);
        setchatUsername('');
      }
    });
  }
  var myUsername = { userUserName };

  // notifications
  const [notification, setnotification] = useState([]);
  useEffect(() => {
    setInterval(() => {
      axios.post('http://localhost:3000/get-notif', myUsername).then(
        (res) => {
          if (res.data.status == "ok") {
            if (res.data.data[0].friends) {
              var notifs = res.data.data[0].friends.filter(function (s) {
                var recieve = s.recieved == true;
                var status = s.status == false;
                return recieve && status;
              });
              // console.log(notifs);
              setnotification(notifs);
            }
          }
        })
    }, 2000)
  }, []);


  var mainnotif = notification.map((S) => {
    return <a key={S.name} href="#" className="filterNotifications all latest
notification" data-toggle="list">
      <img className="avatar-md" src="dist/img/avatars/default.png" data-
toggle="tooltip" data-placement="top" alt="avatar" />
      {/* <div className="status">
        <i className="material-icons online">fiber_manual_record</i>
      </div> */}
      <div className="data">
        <p>{S.name}, has sent you a friend request.</p>
        <button class="btn button col-sm-5 py-2" onClick={() => {
accept(S.name) }} >Accept</button> &nbsp; &nbsp;
        <button class="btn button col-sm-5 py-2" onClick={() => {
decline(myUsername.userUserName, S.name) }} >Decline</button>
      </div>
    </a>
  });
  // Friend list
  const [friendlist, setfriendlist] = useState([]);
  useEffect(() => {
    setInterval(() => {
      axios.post('http://localhost:3000/myFriends', myUsername).then(
        (res) => {
          if (res.data.status == "ok") {
            if (res.data.data[0].friends) {
              var friends = res.data.data[0].friends.filter(function (s) {
                var status = s.status == true;
                return status;
              });
              // console.log(friends);
              setfriendlist(friends);
            }
          }
        })
    }, 2000)
  }, []);


  var friendList = friendlist.map((S) => {
    return <div onClick={() => { openChat(S.name) }} className="filterMembers
all online contact" data-toggle="list" style={{ cursor: "pointer" }}>
      <img className="avatar-md" src="dist/img/avatars/default.png" data-
toggle="tooltip" data-placement="top" title={S.name} alt="avatar" />
      <div className="data">
        <h5>{S.name}</h5>
        {/* <p>will show last message</p> */}
      </div>
      <div className="person-add">
        <i className="material-icons">person</i>
      </div>
    </div>
  });



  const [fname, setfname] = useState([]);
  const [chatName, setchatName] = useState("");
  const [chatProfile, setchatProfile] = useState("");
  const [chatUsername, setchatUsername] = useState("");
  const [sortedMergeMessages, setsortedMergeMessages] = useState([]);
  const [messageList, setmessageList] = useState([]);
  const [messageList2, setmessageList2] = useState([]);
  const [refreshId, setrefreshId] = useState();

  function openChat(fData) {
    setmessage('');
    // console.log(refreshId);
    clearInterval(refreshId);
    axios.post('http://localhost:3000/friendData/?id=' + fData).then(
      (res) => {
        if (res.data.status == "ok") {
          setfname(res.data.data);
          setchatName(res.data.data[0].uname);
          // console.log(chatName);
          setchatUsername(res.data.data[0].uusername);
          setchatProfile(res.data.data[0].profile);
        }
      }
    )

    setrefreshId(setInterval(() => {
      // show friend msg
      axios.post('http://localhost:3000/messages2', { fData }).then(
        (res) => {
          if (res.data.status == "ok") {
            var chat2 = res.data.data.filter(function (s) {
              var friend2 = s.friendUsername === myUsername.userUserName;
              return friend2;
            });
            // console.log(chat2);
            setmessageList2(chat2);
          }
          else {
            setmessageList2([]);
          }
        }
      )

      // Show My Messages
      axios.post('http://localhost:3000/messages1', myUsername).then(
        (res) => {
          if (res.data.status == "ok") {
            var chat = res.data.data.filter(function (s) {
              var friend = s.friendUsername === fData;
              return friend;
            });
            // console.log(chat);
```

```
            setmessageList(chat);
          }
          else {
            setmessageList([]);
          }
        }
      )

    }, 300))
  }

  useEffect(() => {
    var mergeMessages = [...messageList2, ...messageList];
    // console.log(mergeMessages);
    function msgSort(a, b) {
      var frst = new Date(a.dateTime);
      var scnd = new Date(b.dateTime);
      return frst - scnd;
    }
    setsortedMergeMessages(mergeMessages.sort(msgSort));
    // console.log(sortedMergeMessages);

  }, [messageList, messageList2])


  // auto scroll messages
  const messagesEndRef = useRef(null);
  useEffect(() => {
    messagesEndRef.current?.scrollIntoView({ behavior: 'smooth', inline:
'nearest' });
  }, [sortedMergeMessages.length])

  // send messages
  function sendMessage(friendUsername) {
    if (message != '') {
      var today = new Date();
      var date = today.getDate() + '-' + (today.getMonth() + 1) + '-' +
today.getFullYear();
      var time = today.getHours() + ":" + today.getMinutes() + ":" +
today.getSeconds();
      var dateTime = today;
      var messageid = Math.random();

      var sendMessage = { userUserName, friendUsername, message, time, date,
dateTime, messageid };
      axios.post('http://localhost:3000/send-message', sendMessage).then((res)
=> {
        // alert(res.data.data);
      })
      setmessage('');
    }
  }


  // emoji picker
  const [showPicker, setShowPicker] = useState(false);
  const onEmojiClick = (event, emojiObject) => {
    setmessage(prevInput => prevInput + emojiObject.emoji);
    setShowPicker(false);
  };


  // delete a message
  function deleteMsg(id) {
    axios.post('http://localhost:3000/delete-a-message', { id }).then(
      (res) => {
        alert(res.data.data);
      })
  }

  // delete chat history
  function deleteHistory(mine, frnd) {
    axios.post('http://localhost:3000/delete-all-message', { mine, frnd
}).then(
      (res) => {
        alert(res.data.data);
      })
  }




  return (
    <main>
      <div className="layout">
        {/* Start of Navigation */}
        <div className="navigation">
          <div className="container">
            <div className="inside">
              <div className="nav nav-tab menu">

                <button className="btn"><img className="avatar-xl"
src={userImage ? `http://localhost:3000/${userImage}` :
"dist/img/avatars/default.png"} alt="avatar" /></button>
                <a href="#members" className="active" data-toggle="tab"><i
className="material-icons">people</i></a>
                <a href="#notifications" data-toggle="tab"><i
className="material-icons">person_add</i></a>
```

```
                <div data-toggle="tab" className="f-grow1"></div>
                <a href="#settings" data-toggle="tab"><i className="material-
icons">settings</i></a>
              </div>
            </div>
          </div>
        </div>
        {/* End of Navigation */}
        {/* <button onClick={notification}>Check notification</button> */}
        {/* Start of Sidebar */}
        <div className="sidebar" id="sidebar">
          <div className="container">
            <div className="col-md-12">
              <div className="tab-content">
                {/* Start of Contacts */}
                <div className="tab-pane fade active show" id="members">
                  <div className="search">
                    <form className="form-inline position-relative">
                      <input type="search" className="form-control"
id="people" placeholder="Search for people..." />
                      <button type="button" className="btn btn-link loop"><i
className="material-icons">search</i></button>
                    </form>
                    {/* <button className="btn create" data-toggle="modal"
data-target="#exampleModalCenter"><i className="material-
icons">person_add</i></button> */}
                  </div>
                  <div className="contacts">
                    <h1>Chats</h1>
                    {friendList.length != 0 && <div className="list-group"
id="contacts" role="tablist">
                      {friendList}

                    </div>}
                    {friendList.length == 0 && <div className="list-group"
id="contacts" role="tablist" style={{ height: '78vh', fontFamily: 'Seoge UI',
fontSize: '20px', fontWeight: '400', display: 'flex', justifyContent:
'center', flexDirection: 'column', alignItems: 'center' }}>
                      <p style={{ textAlign: "center" }}> Start Chatting to
friends by sending them friend requests. </p>
                    </div>}
                  </div>
                </div>
                {/* End of Contacts */}

                {/* Start of Notifications */}
                <div id="notifications" className="tab-pane fade">
                  <div className="search">
```

```
                    <form className="form-inline position-relative">
                      <input type="search" className="form-control"
id="notice" placeholder="Enter Username to add friends..." />
                      <button type="button" className="btn btn-link loop"><i
className="material-icons">person_add</i></button>
                    </form>
                    <button className="btn create" data-toggle="modal" data-
target="#exampleModalCenter"><i className="material-
icons">person_add</i></button>
                  </div>
                  <div className="notifications">
                    <h1>Friend requests</h1>
                    {mainnotif.length != 0 && <div className="list-group"
id="alerts" role="tablist">
                      {mainnotif}

                    </div>}

                    {mainnotif.length == 0 && <div className="list-group"
id="alerts" role="tablist" style={{ height: '78vh', fontFamily: 'Seoge UI',
fontSize: '20px', fontWeight: '400', display: 'flex', justifyContent:
'center', flexDirection: 'column', alignItems: 'center' }}>
                      <p style={{ textAlign: "center" }}>No one sends you a
friend request :( </p>
                    </div>}

                  </div>
                </div>
                {/* End of Notifications */}
                {/* Start of Settings */}
                <div className="tab-pane fade" id="settings">
                  <div className="settings">
                    <div className="profile">
                      {/* <img className="avatar-xl"
 src="dist/img/avatars/default.png" alt="avatar" /> */}
                      <img className="avatar-xl" src={userImage ?
`http://localhost:3000/${userImage}` : "dist/img/avatars/default.png"}
alt="avatar" />
                      <h1><a href="#">{userName}</a></h1>
                      <span>{userUserName}</span>
                    </div>
                    <div className="categories" id="accordionSettings">
                      <h1>Settings</h1>
                      {/* Start of My Account */}
                      <div className="category">
                        <a href="#" className="title collapsed"
id="headingOne" data-toggle="collapse" data-target="#collapseOne" aria-
expanded="true" aria-controls="collapseOne">
```

```
                          <i className="material-icons md-30
online">person_outline</i>
                          <div className="data">
                            <h5>My Account</h5>
                            <p>Update your profile details</p>
                          </div>
                          <i className="material-
icons">keyboard_arrow_right</i>
                        </a>
                        <div className="collapse" id="collapseOne" aria-
labelledby="headingOne" data-parent="#accordionSettings">
                          <div className="content">
                            <form>
                              <div className="upload">
                                <div className="data">
                                  <img className="avatar-xl" src={userImage ?
`http://localhost:3000/${userImage}` : "dist/img/avatars/default.png"}
alt="image" />
                                  <label>
                                    <input type="file"
accept="image/png,image/jpg,image/jpeg" onChange={(e) => { setProfile(e) }} />
                                    <span className="btn button mr-
3">Upload</span>  {/* {uploadPercentage} %uploaded */}
                                  </label>
                                  <button type="button" className="btn button
w-50 ml-3 bg-green" onClick={updateProfile}>Set</button>
                                </div>
                                <p>For best results, use an image at least
256px by 256px in either .jpg or .png format!</p>
                              </div>
                              {/* <form> */}
                              <div className="parent">
                                <div className="field">
                                  <label htmlFor="Name">Name
<span>*</span></label>
                                  <input type="text" name="Uname"
value={uname} onChange={(e) => { setValue(e) }} className="form-control"
id="Name" placeholder="Name" required />
                                </div>
                                <div className="field">
                                  <label
htmlFor="username">username<span>*</span></label>
                                  <input type="text" name="Uusername"
value={uusername} onChange={(e) => { setValue(e) }} className="form-control"
id="username" placeholder="username" disabled />
                                </div>
                              </div>
                              <div className="field">
```

```
                                <label htmlFor="email">Email
<span>*</span></label>
                                <input type="email" name="Uemail"
value={uemail} onChange={(e) => { setValue(e) }} className="form-control"
id="email" placeholder="Enter your email address" required />
                              </div>
                              <button type="button" className="btn button w-
100" onClick={updateDetails} >Apply Changes</button>
                            </form>

                            <details className='text-dark' style={{ marginTop:
'2rem', fontSize: '1.2rem', fontWeight: 'bold' }}>
                              <summary style={{ fontSize: '1em' }}>Change
Password</summary>
                              <form>
                                <div className="field">
                                  <label htmlFor="password">Old
Password</label>
                                  <input type="password" name="oldpassword"
value={oldpassword} onChange={(e) => { setValue(e) }} className="form-control"
id="password" placeholder="Enter a new password" required />
                                </div>
                                <div className="field">
                                  <label htmlFor="password">New
Password</label>
                                  <input type="password" name="newpassword"
value={newpassword} onChange={(e) => { setValue(e) }} className="form-control"
id="password" placeholder="Enter a new password" required />
                                </div>
                                <div className="field">
                                  <label htmlFor="password">Re-enter New
Password</label>
                                  <input type="password" name="re-enter-new-
password" value={reenternewpassword} onChange={(e) => { setValue(e) }}
className="form-control" id="password" placeholder="Enter a new password"
required />
                                </div>
                                <button type="button" className="btn button w-
100" onClick={changePassword} >Change Password</button>
                              </form>
                            </details>

                          </div>
                        </div>
                      </div>
                      {/* End of My Account */}
                      {/* Start of Appearance Settings */}
                      <div className="category">
```

```
                        <a href="#" className="title collapsed"
id="headingFive" data-toggle="collapse" data-target="#collapseFive" aria-
expanded="true" aria-controls="collapseFive">
                          <i className="material-icons md-30
online">colorize</i>
                          <div className="data">
                            <h5>Appearance</h5>
                            <p>Customize the look of Swipe</p>
                          </div>
                          <i className="material-
icons">keyboard_arrow_right</i>
                        </a>
                        <div className="collapse" id="collapseFive" aria-
labelledby="headingFive" data-parent="#accordionSettings">
                          <div className="content no-layer">
                            <div className="set">
                              <div className="details">
                                <h5>Turn Off Lights</h5>
                                <p>The dark mode is applied to core areas of
the app that are normally displayed as light.</p>
                              </div>
                              <label className="switch">
                                <input type="checkbox" />
                                <span className="slider round mode" />
                              </label>
                            </div>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
                {/* End of Settings */}
              </div>
            </div>
          </div>
        </div>
        {/* End of Sidebar */}
        {/* Start of Add Friends */}
        <div className="modal fade" id="exampleModalCenter" tabIndex={-1}
role="dialog" aria-hidden="true">
          <div className="modal-dialog modal-dialog-centered" role="document">
            <div className="requests">
              <div className="title">
                <h1>Add your friends</h1>
                <button type="button" className="btn" data-dismiss="modal"
aria-label="Close"><i className="material-icons">close</i></button>
              </div>
              <div className="content">
                <form>
                  <div className="form-group">
                    <label htmlFor="user">Username:</label>
                    <input name="friend" value={friend} onChange={(e) => {
setValue(e); }} type="text" className="form-control" id="user"
placeholder="Add recipient..." required />

                  </div>
                  <button type="button" onClick={addFriend} className="btn
button w-100">Send Friend Request</button>
                </form>
              </div>
            </div>
          </div>
        </div>
        {/* End of Add Friends */}

        <div className="main">
          <div className="tab-content" id="nav-tabContent">
            {/* Start of Babble */}
            <div className="babble tab-pane fade active show" id="list-chat"
role="tabpanel" aria-labelledby="list-chat-list">
              {/* Start of Chat */}
              {chatUsername != '' && <div className="chat" id="chat1">
                <div className="top">
                  <div className="container">
                    <div className="col-md-12">
                      <div className="inside">
                        <a href="#"><img className="avatar-md"
src={chatProfile ? `http://localhost:3000/${chatProfile}` :
"dist/img/avatars/default.png"} data-toggle="tooltip" data-placement="top"
title="Keith" alt="avatar" /></a>
                        {/* <div className="status">
                          <i className="material-icons
online">fiber_manual_record</i>
                        </div> */}
                        <div className="data">
                          <h5><a href="#">{chatName}</a></h5>
                          <span>{chatUsername}</span>
                        </div>
                        <button className="btn d-md-block d-none"><i
className="material-icons md-30">info</i></button>
                        <div className="dropdown">
                          <button className="btn" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false"><i className="material-icons md-
30">more_vert</i></button>
                          <div className="dropdown-menu dropdown-menu-right">
```

```
                            <button onClick={() => {
deleteHistory(myUsername.userUserName, chatUsername) }} className="dropdown-
item"><i className="material-icons">delete_forever</i>Clear Chat
History</button>
                            <button onClick={() => {
unfriend(myUsername.userUserName, chatUsername) }} className="dropdown-
item"><i className="material-icons">remove_circle</i>Unfriend
{chatUsername}</button>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
                <div className="content" id="content">
                  <div className="container">

                    {sortedMergeMessages.length != 0 && <div className="col-
md-12 messages">

                      {(() => {
                        return sortedMergeMessages.map((s) => {
                          return <div className={s.friendUsername ==
myUsername.userUserName ? "message" : "message me"}>
                            {s.friendUsername == myUsername.userUserName &&
<img className="avatar-md" src={chatProfile ?
`http://localhost:3000/${chatProfile}` : "dist/img/avatars/default.png"} data-
toggle="tooltip" data-placement="top" alt="avatar" />}
                            <div className="text-main">
                              <div className={s.friendUsername ==
myUsername.userUserName ? "text-group" : "text-group me"}>
                                <div style={{ position: "relative" }}
className={s.friendUsername == myUsername.userUserName ? "text" : "text me"}>
                                  <p>
                                    {myUsername.userUserName ==
s.friendUsername && <span style={{ fontSize: '10px' }}>{s.date}</span>}
                                    {myUsername.userUserName !=
s.friendUsername && <details>
                                      <summary> {s.date}</summary>
                                      <p>
                                        <button onClick={() => {
deleteMsg(s.messageid) }} class="btn button col-sm-5 py-2 text-primary bg-
white" style={{ minWidth: '60px' }}>Delete</button>
                                      </p>
                                    </details>}
                                    {s.message.split('\n').map(str => <p
className='p-0 m-0' style={{ fontSize: '16px', fontWeight: '400', lineHeight:
'1.4', minHeight: '18px' }}>
```

```
                                                {str}
                                              </p>)}
                                            </p>
                                          </div>
                                        </div>
                                        <span>{s.time}</span>
                                      </div>
                                    </div>
                                  })
                                })()}
                                <div ref={messagesEndRef}></div>

                                <div className="picker-container" style={{ position: 
'relative', width: '100%' }}>
                                  {showPicker && <Picker pickerStyle={{ width: '100%', 
height: '30vh' }} onEmojiClick={onEmojiClick} />}
                                </div>
                              </div>}
                              {sortedMergeMessages.length == 0 && <div className="col-
md-12">
                                <div className="no-messages">
                                  <i className="material-icons md-48">forum</i>
                                  <p>Seems people are shy to start the chat. Break the 
ice send the first message.</p>
                                </div>
                                <div className="picker-container" style={{ position: 
'relative', width: '100%' }}>
                                  {showPicker && <Picker pickerStyle={{ width: '100%', 
height: '30vh' }} onEmojiClick={onEmojiClick} />}
                                </div>
                              </div>}
                            </div>
                          </div>

                          <div className="container">
                            <div className="col-md-12">
                              <div className="bottom">
                                <div className="position-relative w-100">
                                  <textarea name="message" value={message} onChange={(e) 
=> { setValue(e); }} className="form-control" placeholder="Start typing..." 
rows={1} defaultValue={""} />
                                  <button className="btn emoticons" onClick={() => 
setShowPicker(val => !val)}><i className="material-
icons">insert_emoticon</i></button>
                                  <button onClick={() => { sendMessage(chatUsername) }} 
className="btn send"><i className="material-icons">send</i></button>
                                </div>
                                <label>
```

```
                        <input type="file" />
                        <span className="btn attach d-sm-block d-none"><i
className="material-icons">attach_file</i></span>
                      </label>
                    </div>
                  </div>
                </div>
              </div>}

              {/* End of Chat */}

              {/* when login ....this div will show on place of chatpage */}
              {chatUsername == '' && <div className="chat" id="chat1" style={{
height: '100vh', backgroundColor: 'lightgrey', display: 'flex',
justifyContent: 'center', flexDirection: 'column', alignItems: 'center' }}>
                <img src="dist\img\homeImg.png" style={{ height: "45vh",
width: '22.5vw', }} />
                <h2 style={{ marginTop: '7vh', color: 'black', fontFamily:
"Seoge UI", fontWeight: '300', lineHeight: '36px', fontSize: '36px' }}>Keep
your device connected</h2>
                <p style={{ marginTop: '2vh', color: 'black', width: '38vw',
fontFamily: "Seoge UI", fontWeight: '400', lineHeight: '22px', fontSize:
'18px', textAlign: 'center' }}>BuddyUp connects to your device to sync
messages. To reduce data usage, connect your device to Wi-Fi.</p>
              </div>}
          </div>
        </div>
      </div> {/* Layout */}
    </main>
  )
}
export default Chatpage
```

Backend

Index.js

```
var express = require('express');
var cors = require('cors');
var bodyParser = require('body-parser');
var MongoClient = require('mongodb').MongoClient;
var ObjectId = require('mongodb').ObjectId;
var nodemailer = require('nodemailer');
var upload = require('./multerConfig');
var path = require('path');
var connectedUsers = [];
var app = express();
app.use(cors());
const server = require('http').Server(app);
const io = require('socket.io')(server);
var client = new
MongoClient("mongodb+srv://buddyupuser:buddyupuser@school.ivhuh.mongodb.net/bu
ddyup?retryWrites=true&w=majority", { useNewURLParser: true,
useUnifiedTopology: true });
var connection;
client.connect((err, db) => {
    if (!err) {
        connection = db;
        console.log("Database Connected Successfully");
    }
    else {
        console.log("Database could not connect");
    }
})
/**
 * SOCKET
 */
io.on('connection', async (socket) => {
    console.log("some client connected");
    var random = Math.random();
    console.log(random);
    connectedUsers.push({ random, socket });
    socket.emit("random", random);
});

app.use(express.static(path.join(__dirname, "userImages")));
// APIs
app.get('/list-account', (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    userCollection.find().toArray((err, docs) => {
        if (!err) {
            res.send({ status: "OK", data: docs })
        }
        else {
            res.send({ status: "Failed", data: err })
        }
    })
});
```

```
app.post('/create-account', bodyParser.json(), (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    userCollection.insert(req.body, (err, result) => {
        if (!err) {
            res.send({ status: "OK", data: "Account Created successfully. You
can login now. You are redirected to login page." })
            sendMail("buddyup28@gmail.com", "kviuqosaxagajcdi",
req.body.uemail, "Welcome to BuddyUp", `<b> Registration successfully </b>
`)

        }
        else {
            res.send({ status: "Failed", data: err })
        }
    })
});

app.post('/valid-email', bodyParser.json(), (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    userCollection.find({ uemail: req.body.uemail }).toArray((err, result) =>
{
        if (!err && result.length > 0) {
            res.send({ status: 'error', data: "this email is already
registered :(" });
        } else {
            res.send({ status: 'ok' })
        }
    })
})

app.post('/valid-username', bodyParser.json(), (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    userCollection.find({ uusername: req.body.uusername }).toArray((err,
result) => {
        if (!err && result.length > 0) {
            res.send({ status: 'error', data: "this username is already
registered :(" });
        } else {
            res.send({ status: 'ok' })
        }
    })
})

app.post("/send-user-otp", bodyParser.json(), (req, res) => {
    console.log(req.body);
    sendMail("buddyup28@gmail.com", "kviuqosaxagajcdi", req.body.uemail,
"Welcome to BuddyUp", `Your One Time Password is -
<h3>${req.body.otp}</h3><br><h6>We hope you find our service cool.</h6>`)
```

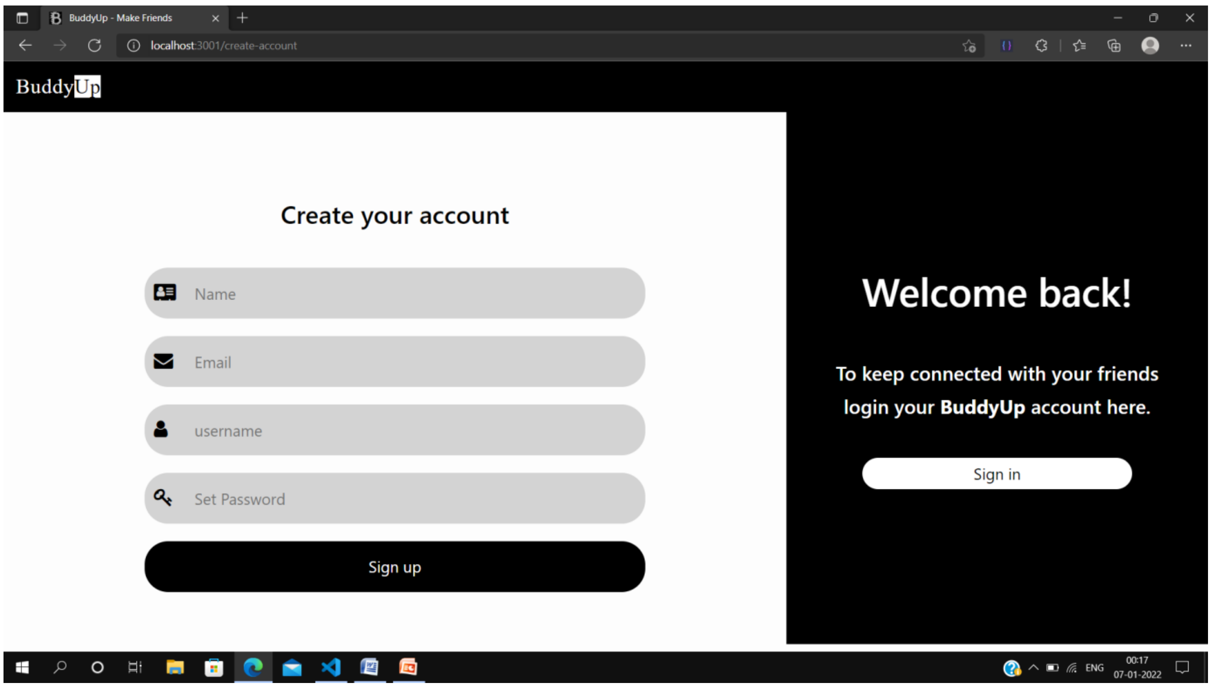
```
    res.send({ status: "ok", data: "please verify your email" });
})

app.post('/check-login', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    var usercollection = connection.db('buddyup').collection('users');
    usercollection.find({ uusername: req.body.username, upassword: 
req.body.password }).toArray((err, result) => {
        if (!err && result.length > 0) {
            res.send({ status: 'ok', data: result[0] });
        } else {
            res.send({ status: 'error', data: err })
        }
    })
})

// for forgot password
app.post('/user-by-email', bodyParser.json(), (req, res) => {
    // console.log("email check");
    // console.log(req.body.email)
    var UserCollection = connection.db('buddyup').collection('users');
    // console.log("var email check three" + req.body.email)
    UserCollection.find({ uemail: (req.body.email) }).toArray((err, result) => 
{
        // console.log("updated student two")
        if (!err && result.length > 0) {
            console.log(result);
            res.send({ status: "ok", data: result })
            // console.log("email is match")
            var n = result.map((e) => { return e.uusername })
            var i = result.map((e) => { return e.upassword })
            sendMail("buddyup28@gmail.com", "kviuqosaxagajcdi", 
req.body.email, "Welcome to BuddyUp", "<h3> your buddyup account  password 
is</h3>" + i + "<h3> your buddyup account  username is </h3>" + n)

        }
        else {
            res.send({ status: "failed", data: err })
        }
    })
})

app.post('/update-user', bodyParser.json(), (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    // console.log(req.body);
    userCollection.update({ _id: ObjectId(req.body.userId) }, { $set: { uname: 
req.body.uname, uemail: req.body.uemail } }, (err, result) => {
        if (!err) {
            res.send({ status: "success", data: "user details updated 
sucessfully" });
        }
        else {
            res.send({ status: "failed", data: err });
        }
    })
})

app.post('/update-password', bodyParser.json(), (req, res) => {
    var userCollection = connection.db('buddyup').collection('users');
    // console.log(req.body);
    userCollection.updateOne({ _id: ObjectId(req.body.userId) }, { $set: { 
upassword: req.body.reenternewpassword } }, (err, result) => {
        if (!err) {
            res.send({ status: "success", data: "Password updated sucessfully" 
});
        }
        else {
            res.send({ status: "failed", data: err });
        }
    })
})

app.post('/update-profile', (req, res) => {
    upload(req, res, (err) => {
        if (err) {
            console.log("Error Occured during upload ");
            console.log(err);
            res.send({ status: "failed", data: err });
        }
        else {
            var userCollection = connection.db('buddyup').collection('users');
            console.log(req.files);
            // console.log(req.files.profile[0].filename);
            // console.log(req.body._id);
            userCollection.update({ _id: ObjectId(req.body._id) }, { $set: { 
profile: req.files.profile[0].filename } }, (err, result) => {
                if (!err) {
                    res.send({ status: "success", data: "Profile updated 
sucessfully" });
                }
                else {
                    res.send({ status: "failed", data: err });
                }
            })
        }
    })
```

```
})

app.post('/add-friend', bodyParser.json(), (req, res) => {

    const collection = connection.db('buddyup').collection('users');
    var friend = req.body.friend;
    // console.log(friend);
    var myUsername = req.body.userUserName;
    console.log(myUsername);
    console.log(req.body);
    collection.updateOne({ 'uusername': myUsername }, { $push: { friends: {
name: friend, status: false, sent: true, recieved: false } } })
    collection.updateOne({ 'uusername': friend }, { $push: { friends: { name:
myUsername, status: false, sent: false, recieved: true } } }
        , (err, result) => {
            if (!err) {
                res.send({ status: "ok", data: "Friend Request Sent" });
            }
            else {
                res.send({ status: "failed", data: "some error occured" });
            }
        })
});

app.post('/search-for-user', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection = connection.db('buddyup').collection('users');
    collection.find({ uusername: (req.body.friend) }).toArray((err, docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

app.post('/get-notif', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection = connection.db('buddyup').collection('users');
    collection.find({ uusername: (req.body.userUserName) }).toArray((err,
docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

app.post('/get-userlist', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection = connection.db('buddyup').collection('users');
    collection.find({ uusername: (req.body.S) }).toArray((err, docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

app.post('/accept-request', bodyParser.json(), (req, res) => {

    const collection = connection.db('buddyup').collection('users');
    var friend = req.body.friendReq;
    var username = req.body.userUserName;
    // console.log(friend); console.log(username);
    collection.updateOne({ "uusername": username, "friends": { $elemMatch: {
"name": friend } } }, { $set: { "friends.$.status": true } })
    collection.updateOne({ "uusername": friend, "friends": { $elemMatch: {
"name": username } } }, { $set: { "friends.$.status": true } }
        , (err, result) => {
            if (!err) {
                res.send({ status: "ok", data: "friend request accepted" });
            }
            else {
                res.send({ status: "failed", data: "some error occured" });
            }
        })

});

app.post('/unfriend-or-decline', bodyParser.json(), (req, res) => {
    const collection = connection.db('buddyup').collection('users');
    console.log(req.body);
uusername": req.body.mine, "friends": { $elemMatch: { "name": req.body.frnd }
} }, { $set: { "friends.$.name": "", "friends.$.status": false,
"friends.$.recieved": false, "friends.$.sent": false } })
    collection.updateOne({ "uusername": req.body.frnd, "friends": {
$elemMatch: { "name": req.body.mine } } }, { $set: { "friends.$.name": "",
"friends.$.status": false, "friends.$.recieved": false, "friends.$.sent":
false } }
        , (err, result) => {
            if (!err) {
                res.send({ status: "ok", data: "friend unfriend or request
declined" });
            }
            else {
                res.send({ status: "failed", data: "some error occured" });
            }
        })
});

app.post('/myFriends', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection = connection.db('buddyup').collection('users');
    collection.find({ uusername: (req.body.userUserName) }).toArray((err,
docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

app.post('/friendData', bodyParser.json(), (req, res) => {
    // console.log("friend data = " + req.query.id);
    const collection = connection.db('buddyup').collection('users');
    collection.find({ uusername: (req.query.id) }).toArray((err, docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

app.post('/send-message', bodyParser.json(), (req, res) => {
    const collection = connection.db('buddyup').collection('messages');
    collection.insertOne(req.body, (err, result) => {
        if (!err) {
            res.send({ status: "ok", data: "Message Sent" });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});
```

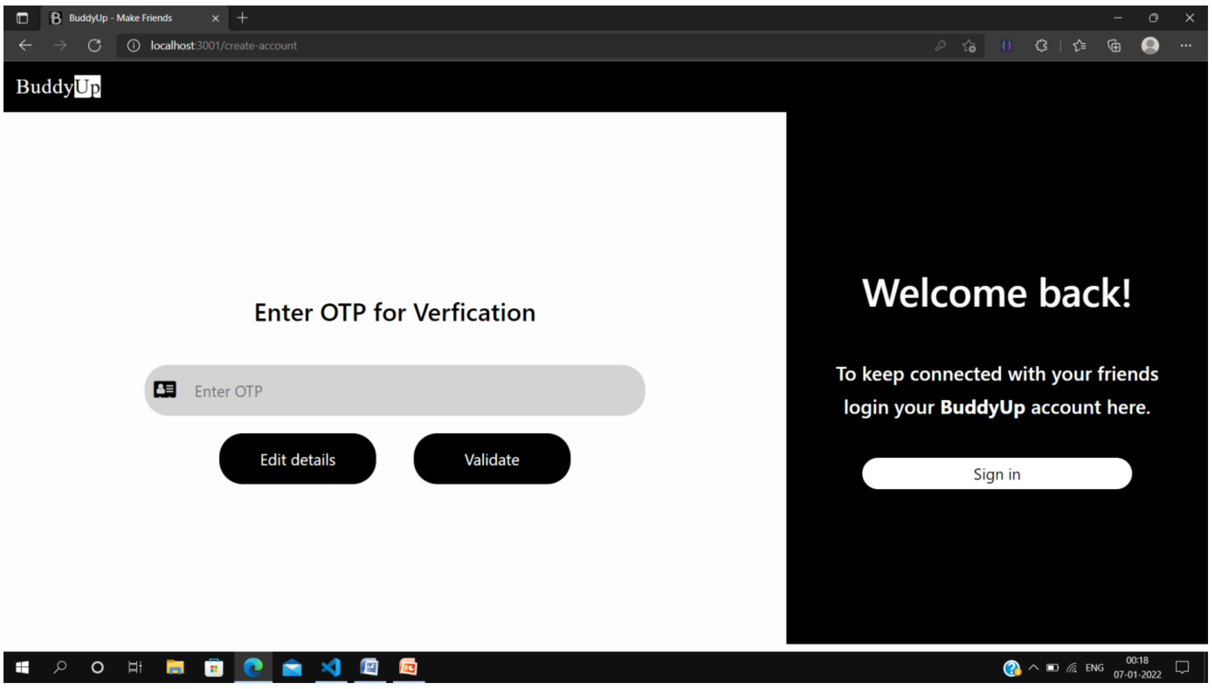
```
// for retrieving messages
app.post('/messages1', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection = connection.db('buddyup').collection('messages');
    collection.find({ userUserName: (req.body.userUserName) }).toArray((err,
docs) => {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

// for retrieving messages
app.post('/messages2', bodyParser.json(), (req, res) => {
    // console.log(req.body);
    const collection2 = connection.db('buddyup').collection('messages');
    collection2.find({ userUserName: (req.body.fData) }).toArray((err, docs)
=> {
        if (!err) {
            res.send({ status: "ok", data: docs });
        }
        else {
            res.send({ status: "failed", data: "some error occured" });
        }
    })
});

// for delete a messages
app.post('/delete-a-message', bodyParser.json(), (req, res) => {
    const collection = connection.db('buddyup').collection('messages');
    // console.log(req.body.id);
    collection.deleteOne({ messageid: (req.body.id) }, (err, result) => {
        if (!err) {
            res.send({ status: "OK", data: "Message Deleted successfully" })
        }
        else {
            res.send({ status: "Failed", data: err })
        }
    })
});

// delete all msg between two persons
app.post('/delete-all-message', bodyParser.json(), (req, res) => {
    const collection = connection.db('buddyup').collection('messages');
    // console.log(req.body);
    collection.deleteMany({ userUserName: (req.body.frnd), friendUsername:
(req.body.mine) })
    collection.deleteMany({ userUserName: (req.body.mine), friendUsername:
(req.body.frnd) }, (err, result) => {
        if (!err) {
            res.send({ status: "OK", data: "All Messages Deleted" })
        }
        else {
            res.send({ status: "Failed", data: err })
        }
    })
});
function sendMail(from, appPassword, to, subject, htmlmsg) {
    let transporter = nodemailer.createTransport(
        {
            host: "smtp.gmail.com",
            port: 587,
            secure: false,
            auth:
            {
                user: from,
                pass: appPassword
            }
        }
    );
    let mailOptions =
    {
        from: from,
        to: to,
        subject: subject,
        html: htmlmsg
    };
    transporter.sendMail(mailOptions, function (error, info) {
        if (error) {
            console.log(error);
        }
        else {
            console.log('Email sent:' + info.response);
        }
    });
}
server.listen(3000, () => {
    console.log("Server is started on port 3000");
});
```

SCREENSHOTS

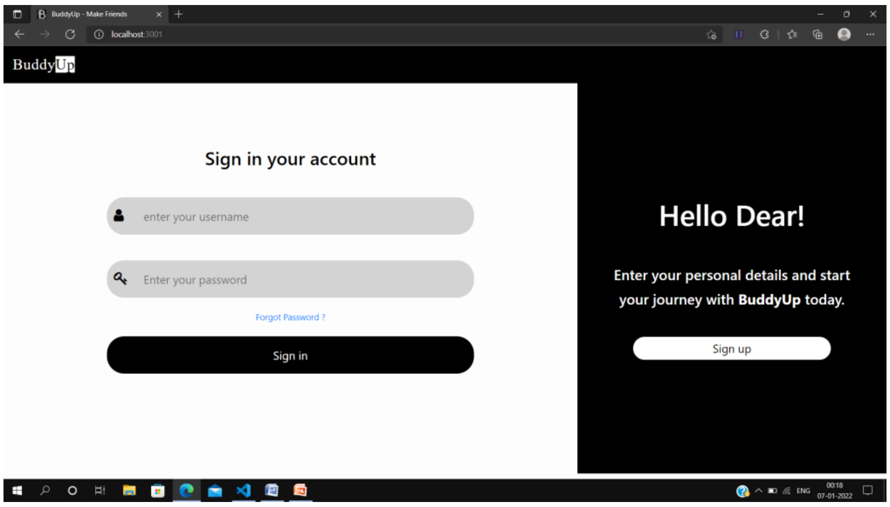
Signup:



OTP for validation at time of signup:


BuddyUp - Make Friends
localhost:3001/create-account
BuddyUp
Enter OTP for Verfication
Enter OTP
Edit details
Validate
Welcome back!
To keep connected with your friends login your **BuddyUp** account here.
Sign in
00:18
07-01-2022

Signin:



BuddyUp

Sign in your account

enter your username

Enter your password

Forgot Password ?

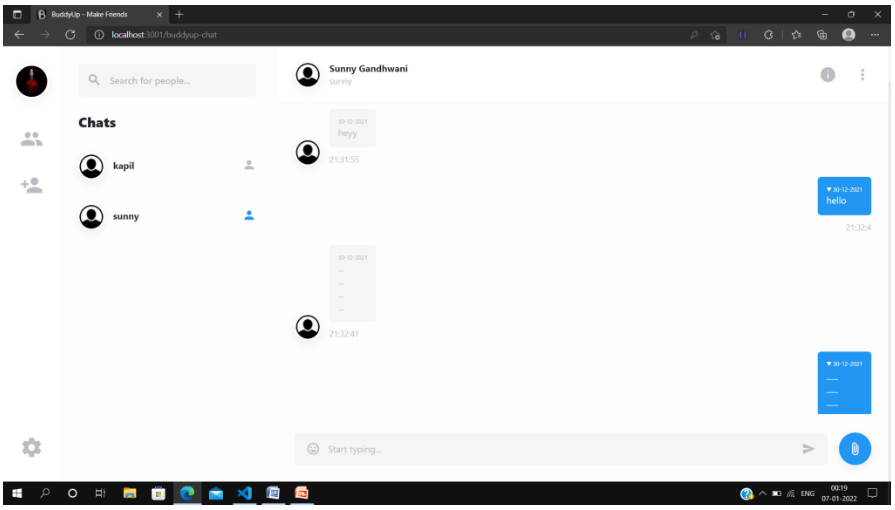
Sign in

Hello Dear!

Enter your personal details and start your journey with **BuddyUp** today.

Sign up

Chat Page:



Search for people...

Chats

kapil

sunny

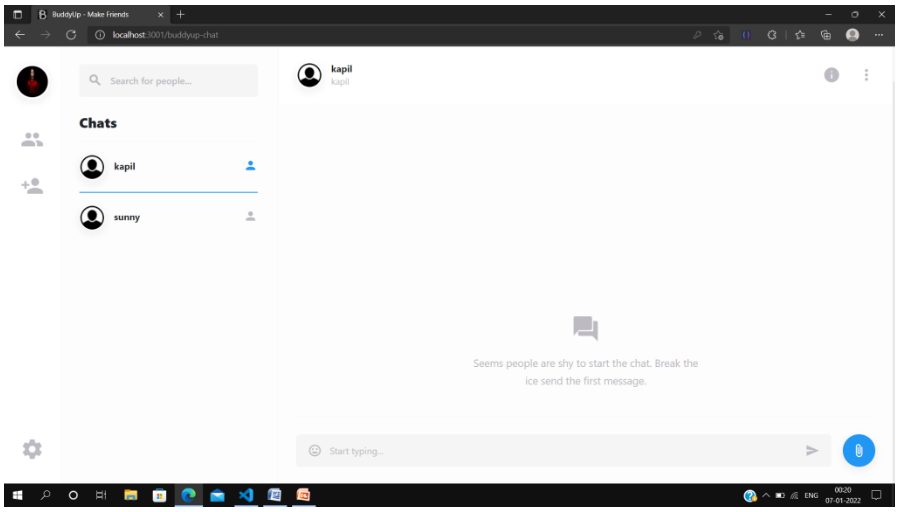
Sunny Gandhwani
sunny

30-12-2021
heyy
21:31:55

30-12-2021
hello
21:32:4

21:32:41

Start typing...



Search for people...

Chats

kapil

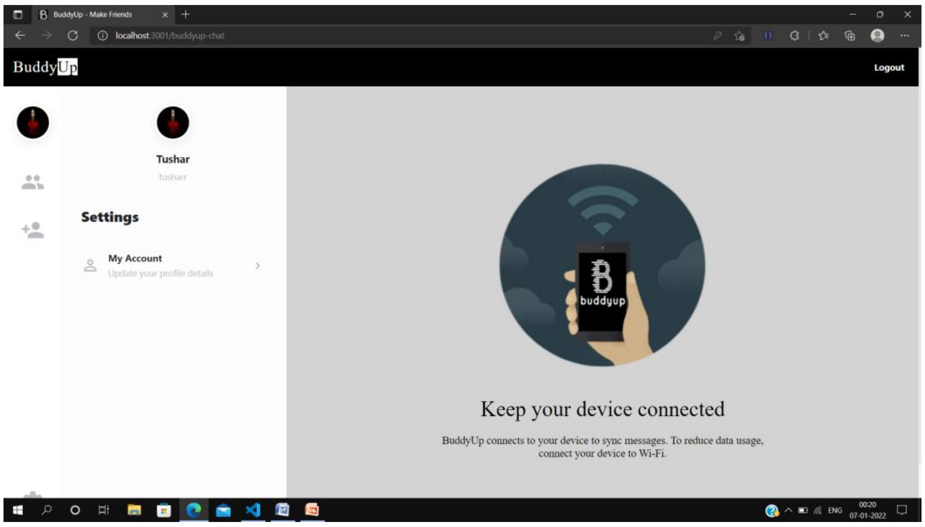
sunny

kapil
kapil

Seems people are shy to start the chat. Break the ice send the first message.

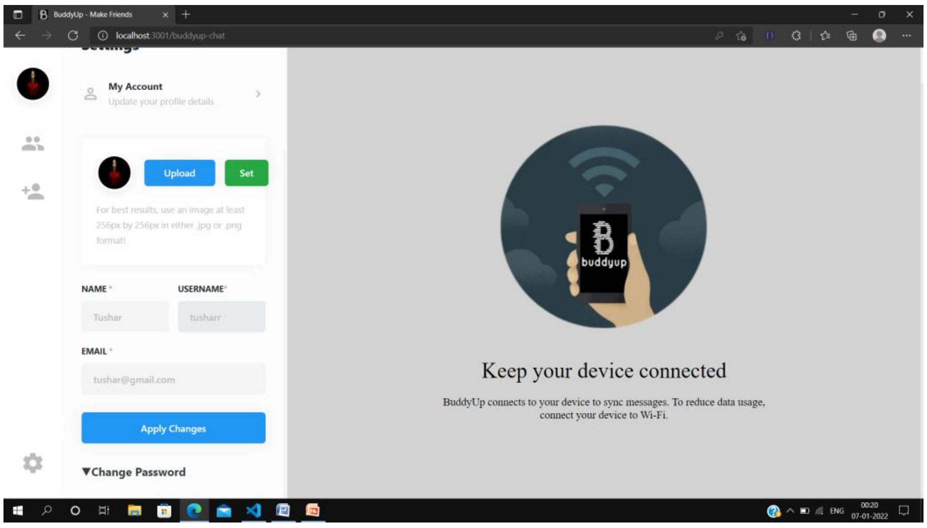
Start typing...

Settings:



Keep your device connected

BuddyUp connects to your device to sync messages. To reduce data usage, connect your device to Wi-Fi.

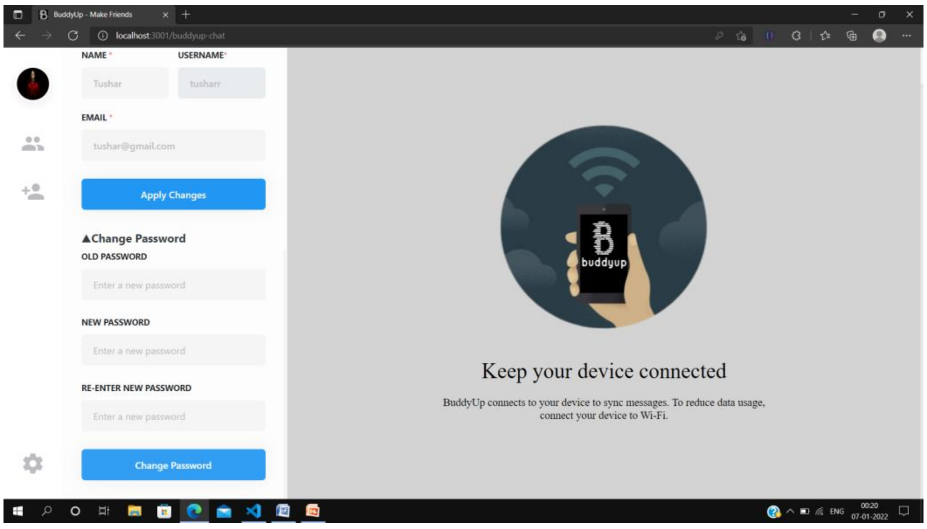


For best results, use an image at least 256px by 256px in either .jpg or .png format!

NAME: Tushar
USERNAME: tusharr
EMAIL: tushar@gmail.com

Keep your device connected

BuddyUp connects to your device to sync messages. To reduce data usage, connect your device to Wi-Fi.



NAME: Tushar
USERNAME: tusharr
EMAIL: tushar@gmail.com

Keep your device connected

BuddyUp connects to your device to sync messages. To reduce data usage, connect your device to Wi-Fi.

References

https://github.com/

https://stackoverflow.com/

List of Symbols, Abbreviations and Nomenclature

MongoDB	Non relational database
React	Javascript frontend library
Node	Javascript runtime environment
Express	Javascript library for creating APIs.
CSS	Cascading Style Sheets
JSX	JavaScript XML
OS	Operating System
RAM	*Random-access memory*
Socket	Javascript library used to real time chat feature