PROJECT

CHAT APPLICATION USING REACT JS

ABSTRACT

Chat refers to the process of communicating, interacting and/or exchanging messages over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audio or video communication via the Internet A chat application has basic two components, viz server and client. A server is a computer program or a device that provides functionality for other programs or devices. Clients who want to chat with each other connect to the server The chat application we are going to make will be more like a chat room, rather than a peer to peer chat. So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user . With the rapid development of mobile phones, mobile devices have become one of the integral part of daily activities. In recent years, chat applications have evolved and made a major change in social media because of their distinctive features that attract audiences. It provides real-time messaging and offers different services including, exchange text messages, images, files and etc. Moreover, it supports cross platforms such as Android and iOS. There are currently hundred millions of users smartphone are using chat applications on monthly basis. There are two types of architecture in those applications, client-server and peer-to-peer networks. In a peer-to-peer network, there is no central server and each user has his/her own data storage. On the contrary, there are dedicated servers and clients in a client-server network and the data is stored on a central server[3]. Security and privacy in chat applications have a paramount importance but few people take it seriously. In a test done by the Electronic Frontier Foundation, most of the popular messaging applications failed to meet most security standards. These applications might be using the conversations as an information for certain purposes. Moreover, reading the private conversations is certainly unacceptable in terms of privacy. Most applications only used Transport Layer Security (TLS) for securing channel, the service provider has full access to every message exchanged through their infrastructure . Therefore, these messages can be accessed by attackers. Therefore to maintain protection and privacy, messages should be encrypted from sender to receiver and no one can read messages even the service provider, in addition to protecting the local storage of the device . In this paper, we focus on security, privacy and speed by proposing end-to-end security which ensures only sender and receiver can read messages without a third party. As well as storage protection and fast transfer of messages between the parties. The main contributions of this paper are the following: 1- Propose client-server mobile chat application which supports the status of the communicating parties whether online or offline. 2- Provide a friendship request service. 3. Secure key exchange, then calculate the session key. 4. Secure exchange of end-to-end message 5.Analysis and test the proposed chat..

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

API	**Application Programming Interface**
JSON	**JAVA SCRIP OBJECT NOTATION**
JS	**JAVA SCRIPT**
CSS	**CASCADING STYLES SHEET**
HTML	**HYPER TEXT MARKUP LANGUAGE**

AIM:

To create a chat Application using React js

DESCRIPTION

A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time. With a chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person. This also keeps users conversing on your platform instead of looking elsewhere for a messaging solution. Whether it’s a private chat, group chat, or large scale chat, adding personalized chat features to your app can help ensure that your users have a memorable experience.

.

BACKEND - REACTJS

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

.

PROJECT OUTLINE

User can login to the chat application using their username and password then allowed to use the chat app. In this chat application user can send and receive text Messages, Multimedia , create groups or chat groups for their common discussion . chat admin is someone who creates the group who’s allowed to add any numbers user under the group where they can share their common thoughts. After using the chat application user can logout the chat app through the logout option.

SCOPE AND OBJECTIVE

The chat application is built using react js deployed using chat engine api , and hosted on netlify. The main aspect is where one user can can send text ,images from one end and the other receives the same from their end. Also user can create a chat group ,add any number of people to their wish and share the text,images apparently the other members of the group receives the same in the chat room.

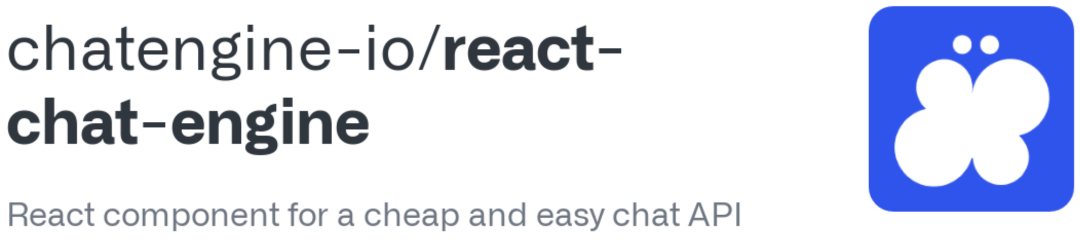
HARDWARE REQUIREMENTS

- Windows system
- A good internet quality

SOFTWARE REQUIREMENTS

- react js
- chat engine api
- netlify to host chat app

CHAT ENGINE– API:



chatengine-io/**react-chat-engine**

React component for a cheap and easy chat API

API stands for Application Programming Interface. An **API** is a software intermediary that allows two applications to talk to each other **APIs unlock a door to software (or web-based data)**, in a way that is controlled and safe for the program. Code can then be entered that sends requests to the receiving software, and data can be returned. A clear example of this in action is the Google Maps API

Chat Engine is an API which makes it easy to build chat services. Building a chat from scratch takes a lot of time, code, and is expensive. It's better to use a product instead of writing it from scratch. We make it easy to build your chat idea in minutes.

FEATURES:

- Authenticate user
- Subscribe (connect) to incoming chats and messages
- Create chats and messages
- Add and remove people from chats
- Edit and delete chat and message data.

CHAT OBJECT:

- id (int) - Unique primary key to identify this chat
- admin (String) - Unique username of the person who created this chat
- title (String) - Optional title of this chat
- created (Datetime) - Date-time of chat creation
- people (Array) - Array of people added to this chat

MESSAGE OBJECT:

-
- id (int) - Unique primary key to identify this message
- sender (String) - Unique username of the person who sent this message
- text (String) - Contents of the message sent
- created (Datetime) - Date-time of message creation

WORKING OF API KEY

An application programming interface key (API key) is a unique code that is passed in to an API to identify the calling application or user. API keys are used to track and control how the API is being used, for example to prevent malicious use or abuse of the API. The API key often acts as both a unique identifier and a secret token for authentication, and is assigned a set of access that is specific to the identity that is associated with it.

NETLIFY

Netlify is a web developer platform that multiplies productivity By unifying the elements of the modern decoupled web, from local development to advanced edge logic, Netlify enables a 10x faster path to much more performance, secure, and scalable websites and apps. Our bet on the Jamstack is quickly coming true. The web is rapidly changing away from monolithic to decoupled apps, and web developers are storming ahead with more power than ever. Netlify is built to cater to that movement, and in just a few years we've on-boarded millions of developers and businesses, and are building and serving millions of web projects daily around the globe. Fun fact: in the time it took you to read the above, Netlify served over 600,000 requests.

ADVANTAGES:

- Netlify Is Less Expensive, and You Get a Faster Site.
- Netlify Build Enables Developers to Build With Any Integration.
- It's Easier to Launch a Site Using Netlify.

DISADVANTAGES:

- Netlify Is Less Expensive, and You Get a Faster Site.
- Netlify Build Enables Developers to Build With Any Integration.
- It's Easier to Launch a Site Using Netlify.

HTML

The **Hyper Text Mark-up Language** or **HTML** is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading StyleSheets(CSS) and scripting languages such

as JavaScript
HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark up language such as HTML CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript

CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .

REACT-JS

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

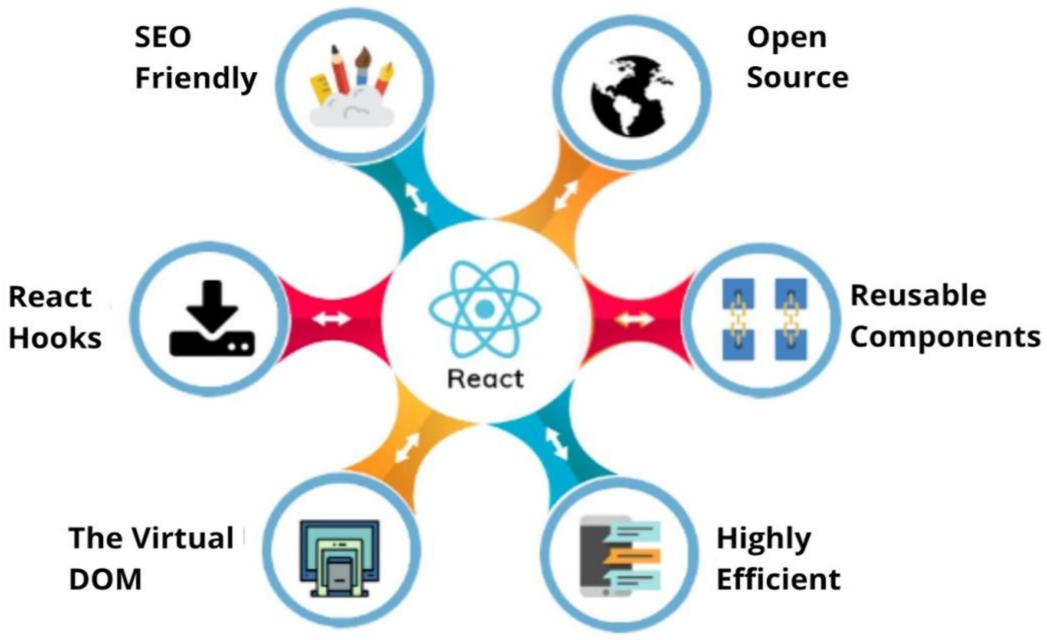
React js requirements:

- HTML + CSS. No front-end dev is a stranger to HTML and CSS. ...
- JSX. In React, you never really touch HTML proper. ...
- JavaScript Fundamentals + ES6. ...
- Node + npm. ...
- Redux.

NEEDS OF REACT-JS

To react, you just need basic knowledge of CSS and HTML. React can be used to create mobile applications (React Native). And React is a diehard fan of reusability, meaning extensive code reusability is supported. So at the same time, we can make IOS, Android and Web applications.
React is an excellent tool with which to create interactive applications for mobile, web, and other platforms. React's popularity and usage are increasing day by day for good reason. As a developer, coding in React makes you better at JavaScript, a language that holds nearly 90% of the web development share today.

ADVANTAGES:

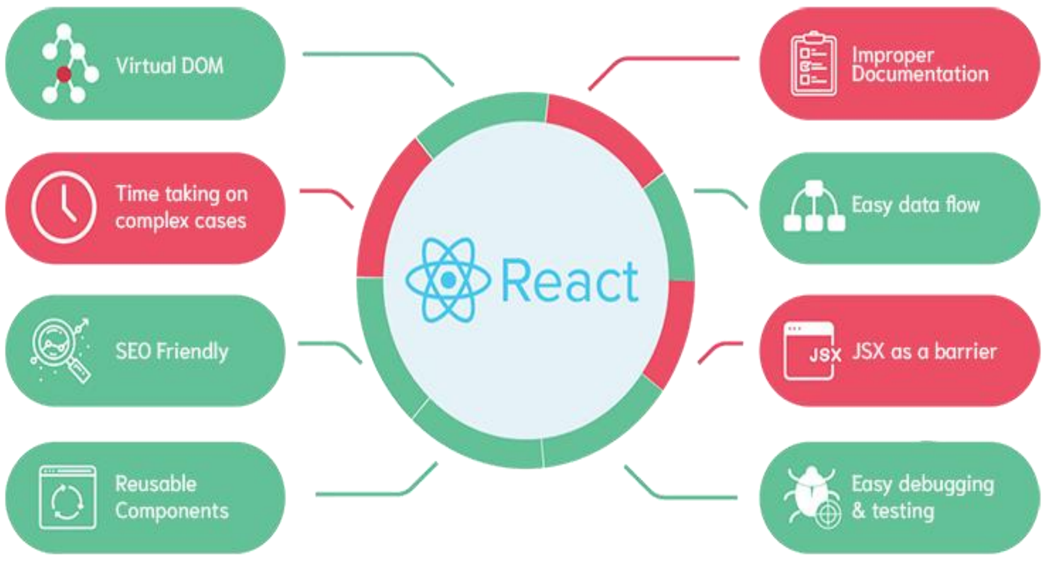


- Makes JavaScript coding easier.
- Extremely competent.
- Excellent cross-platform support.
- Handles dependencies.
- Template designing made easy.
- Provides amazing developer tools.
- UI focused designs.
- Easy to adopt.

DISADVANTAGES:

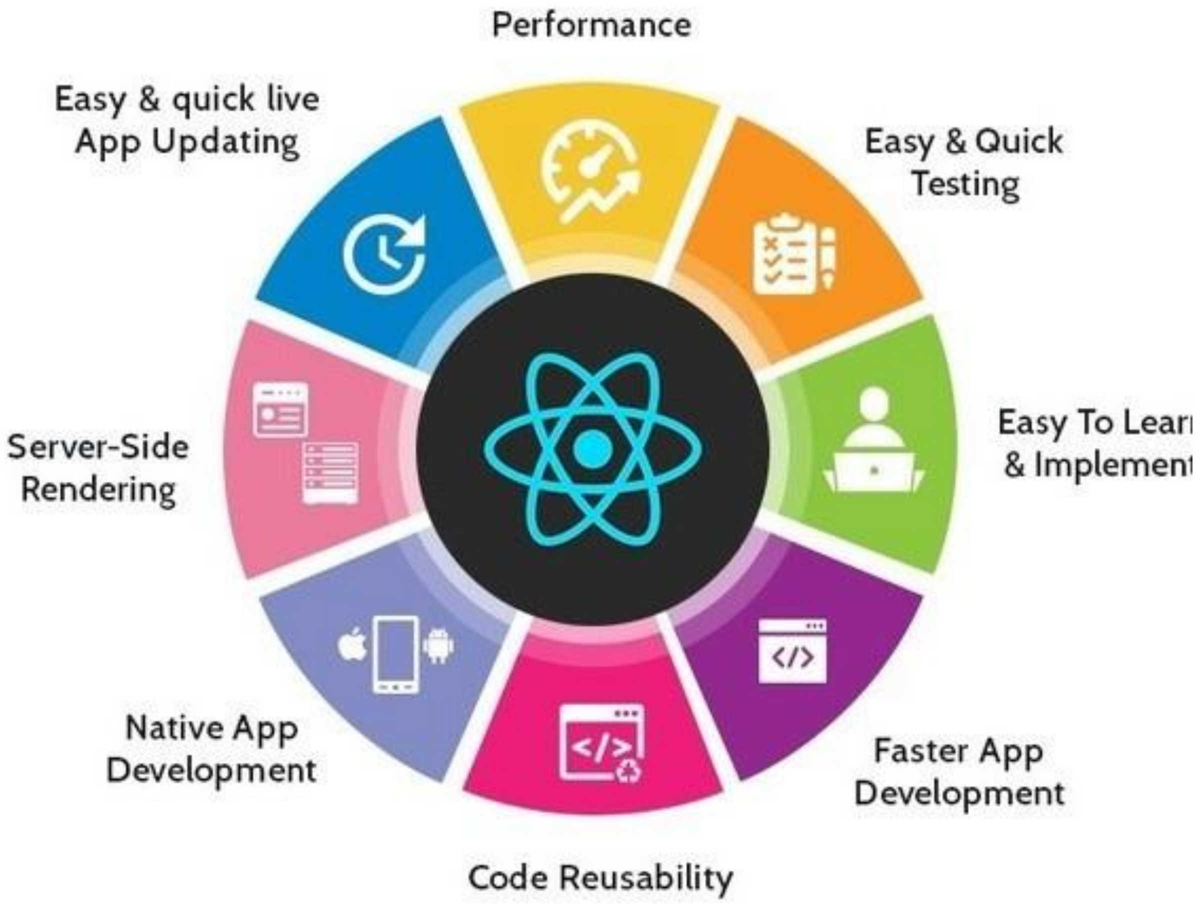
- The high pace of development. The high pace of development has an advantage and disadvantage both. ...
- Poor Documentation. It is another cons which are common for constantly updating technologies. ...
- View Part. ReactJS Covers only the UI Layers of the app and nothing else. ...
- JSX as a barrier.

PROS AND CONS

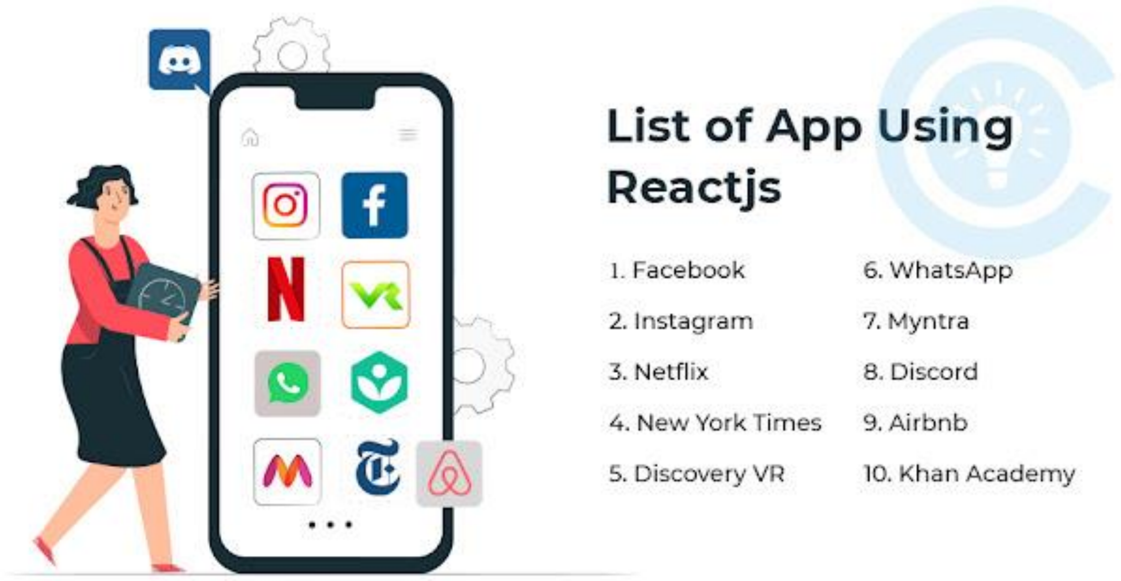


BENIFITS OF REACT-JS

- Performance
- Code reusability
- Easy to learn easy and quick testing
- Native app development
- Server side rendering



APPLICATIONS OF REACT-JS



List of App Using Reactjs

1. Facebook
2. Instagram
3. Netflix
4. New York Times
5. Discovery VR
6. WhatsApp
7. Myntra
8. Discord
9. Airbnb
10. Khan Academy

SOURCE CODE:

App.css

```
* {
  font-family: Avenir, -apple-system, BlinkMacSystemFont, Segoe UI, Roboto,
    Helvetica Neue, Arial, Noto Sans, sans-serif, Apple Color Emoji,
    Segoe UI Emoji, Segoe UI Symbol, Noto Color Emoji;
  letter-spacing: 0.5px;
}

.ce-chat-list {
  background-color: rgb(240, 240, 240) !important;
}

.ce-chats-container {
  background-color: rgb(240, 240, 240) !important;
}

.ce-active-chat-card {
  background-color: #cabcdc !important;
  border: 4px solid #cabcdc !important;
  border-radius: 0px !important;
}

.ce-chat-subtitle-text {
  color: #595959 !important;
}
```

```
.ce-chat-form-container {
  padding-bottom: 20px !important;
}

.ce-text-input {
  border-radius: 6px !important;
  border: 1px solid #3b2a50 !important;
}

.ce-primary-button {
  border-radius: 6px !important;
  background-color: #7554a0 !important;
  position: relative;
  bottom: 1px;
}

.ce-danger-button {
  background-color: white !important;
  border-radius: 22px !important;
}

.ce-settings {
  background-color: rgb(240, 240, 240) !important;
}

.ce-autocomplete-input {
  border-radius: 6px !important;
  border: 1px solid #3b2a50 !important;
}

.ce-autocomplete-options {
  border-radius: 6px !important;
  border: 1px solid #3b2a50 !important;
  background-color: white !important;
}

.ce-chat-settings-container {
  padding-top: 12px !important;
}

.ce-chat-avatars-row {
  display: none !important;
}

/* CUSTOM FEED */

.chat-feed {
  height: 100%;
  width: 100%;
  overflow: scroll;
  background-color: rgb(240, 240, 240);
}
```

```
.chat-title-container {
  width: calc(100% - 36px);
  padding: 18px;
  text-align: center;
}

.chat-title {
  color: #7554a0;
  font-weight: 800;
  font-size: 24px;
}

.chat-subtitle {
  color: #7554a0;
  font-weight: 600;
  font-size: 12px;
  padding-top: 4px;
}

.message-row {
  float: left;
  width: 100%;
  display: flex;
  margin-left: 18px;
}

.message-block {
  width: 100%;
  display: inline-block;
}

.message-avatar {
  width: 44px;
  height: 44px;
  border-radius: 22px;
  color: white;
  text-align: center;
  background-repeat: no-repeat;
  background-position: center;
  background-size: 48px;
}

.logout-button {
  text-align: center;
  margin-top: 20px;
  margin-left: 43%;
  margin-bottom: 40px;
  width: 70px;
  border-radius: 6px !important;
  border: 1px solid #3b2a50 !important;
  background-color: #7554a0 !important;


}
```

```
.message {
  padding: 12px;
  font-size: 16px;
  border-radius: 6px;
  max-width: 60%;
}

.message-image {
  margin-right: 18px;
  object-fit: cover;
  border-radius: 6px;
  height: 30vw;
  /* width: 30vw; */
  max-height: 200px;
  max-width: 200px;
  min-height: 100px;
  min-width: 100px;
}

.read-receipts {
  position: relative;
  bottom: 6px;
}

.read-receipt {
  width: 13px;
  height: 13px;
  border-radius: 13px;
  margin: 1.5px;
  background-repeat: no-repeat;
  background-position: center;
  background-size: 14px;
}

.message-form-container {
  position: absolute;
  bottom: 0px;
  width: calc(100% - 36px);
  padding: 18px;
  background-color: rgb(240, 240, 240);
}

.message-form {
  overflow: hidden;
  border-radius: 6px;
  border: 1px solid #3b2a50;
  background-color: white;
}

.message-input {
  height: 40px;
  width: calc(100% - 132px);
  background-color: white;
  border: 1px solid white;
  padding: 0px 18px;
  outline: none;
  font-size: 15px;
}

.image-button {
  cursor: pointer;
  padding: 0px 12px;
  height: 100%;
}

.send-button {
  height: 42px;
  background-color: white;
  border: 1px solid white;
  padding: 0px 18px;
  cursor: pointer;
}

.send-icon {
  top: 1px;
  position: relative;
  transform: rotate(-90deg);
}

.picture-icon {
  top: 1px;
  position: relative;
  font-size: 14px;
}

/* FORM STYLES */
*,
*::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-size: 62, 5%;
}

.wrapper {
  height: 100vh;
  width: 100%;
  background: rgb(117, 84, 160);
  background: linear-gradient(90deg, rgba(117, 84, 160, 1) 7%, rgba(117, 84, 160, 1) 17%, rgba(106,
95, 168, 1) 29%, rgba(99, 103, 174, 1) 44%, rgba(87, 116, 184, 1) 66%, rgba(70, 135, 198, 1) 83%,
rgba(44, 163, 219, 1) 96%, rgba(22, 188, 237, 1) 100%, rgba(0, 212, 255, 1) 100%);
  display: flex;
  justify-content: center;
  align-items: center;
}

.input {
  color: #333;
  font-size: 1.2rem;
  margin: 0 auto;
  padding: 1.5rem 2rem;
  border-radius: 0.2rem;
  background-color: rgb(255, 255, 255);
  border: none;
  width: 90%;
  display: block;
  border-bottom: 0.3rem solid transparent;
  transition: all 0.3s;
  outline: none;
  margin-bottom: 25px;
}

.form {
  width: 400px;
}

.title {
  text-align: center;
  color: white;
  margin-bottom: 30px;
  width: 100%;
  font-size: 2.3em;
  ;
}
```

```
.button {
  border-radius: 4px;
  border: none;
  background-color: white;
  color: black;
  text-align: center;
  text-transform: uppercase;
  font-size: 22px;
  padding: 20px;
  width: 200px;
  transition: all 0.4s;
  cursor: pointer;
  margin: 5px;
  width: 90%;
}

.button span {
  cursor: pointer;
  display: inline-block;
  position: relative;
  transition: 0.4s;
}

.button span:after {
  content: '\00bb';
  position: absolute;
  opacity: 0;
  top: 0;
  right: -20px;
  transition: 0.5s;
}

.button:hover span {
  padding-right: 25px;
}

.button:hover span:after {
  opacity: 1;
  right: 0;
}

.error {
  color: white;
  text-align: center;
  margin-top: 20px;
}
```

APP.js

```
import { ChatEngine } from "react-chat-engine";

import './App.css';
import ChatFeed from  './components/ChatFeed';
import LoginForm from "./components/LoginForm";

const App = () => {
  if(!localStorage.getItem('username')) return <LoginForm />
  return (
    <ChatEngine
      height = "100vh"
      projectID = "7a704088-98dc-4203-a0f7-cc2159f81d85"
      userName = {localStorage.getItem('username')}
      userSecret = {localStorage.getItem('password')}
      renderChatFeed = {(ChatAppProps) => <ChatFeed {... ChatAppProps} />}
    />
  );
}
export default App;
```

index.js

```
import  React from 'react';
import  ReactDom from  'react-
dom';
import App  from  './App';
ReactDom.render(<App/>,
document.getElementById('root'))
;
```

Chatfeed.jsx

```
import MessageForm from "./MessageForm";
import MyMessage from "./MyMessage";
import TheirMessage from "./TheirMessage";

const ChatFeed = (props) => {
    const { chats, activeChat, userName,
messages } = props;
```

```
    const chat = chats &&
chats[activeChat];

    const renderReadReceipts = (message,
isMyMessage) => {
        return chat.people.map((person,
index) => person.last_read === message.id
&& (
            <div
                key={`read_${index}`}
                className="read-receipt"
                style={
                    {
                        float: isMyMessage
? 'right' : 'left',
                        backgroundImage:
`url(${person?.person?.avatar})`
                    }
                }

            />
        ))
    }

    const renderMessages = () => {
        const keys = Object.keys(messages);
        return keys.map((key, index) => {
            const message = messages[key];
            const lastMessageKey = index
=== 0 ? null : key[index - 1];
            const isMyMessage = userName
=== message.sender.username;

            return (
                <div key={`msg_${index}`}
style={{ width: '100%' }}>
                    <div
className="message-block">
                        {
                            isMyMessage
```

```
                                ?
<MyMessage message={message} />
                                :
<TheirMessage message={message}
lastMessage={messages[lastMessageKey]} />
                        }

                    </div>

                    <div className="read-
receipts" style={{ marginRight: isMyMessage
? '18px' : '0px', marginLeft: isMyMessage ?
'0px' : '68px' }}>

{renderReadReceipts(message, isMyMessage)}
                    </div>
                </div>


            )
        })

    }

    const logout = () => {

localStorage.removeItem('username');

localStorage.removeItem('password');

        window.location.reload();
    }

    if (!chat) return '...loading';

    return (
        <div className="chat-feed">
            <div className="chat-title-
container">
                <div className="chat-
title">
```

```
                    {chat.title}
                </div>
                <div className="chat-
subtitle">

{chat.people.map((person) => `
${person.person.username}`)}
                </div>
                <div className="logout-
button">
                    <button
onClick={logout}>Logout</button>
                </div>



                {renderMessages()}
                <div style={{ height:
'100px' }} />
                <div className="message-
form-container">
                    <MessageForm {...props}
chatId={activeChat} />
                </div>
            </div>
        </div>
    )
}
export default ChatFeed;
```

loginform.jsx

```
import { useState } from "react";
import axios from "axios";

const LoginForm = () => {

    const [username, setUsername] =
useState("");
    const [password, setPassword] =
useState("");
    const [error, setError] = useState("");
    const handleSubmit = async (e) => {
        e.preventDefault();

        const authObject = { 'project-Id':
"7a704088-98dc-4203-a0f7-cc2159f81d85",
'User-Name': username, 'User-Secret':
password }

        try {
            await
axios.get("https://api.chatengine.io/chats"
, { headers: authObject });


localStorage.setItem('username', username);

localStorage.setItem('password', password);

            window.location.reload();

        } catch (error) {
            setError("Oops, Wrong
Credentials..Try again")
        }

    }

    return (
        <div className="wrapper">
            <div className="form">
                <h1 className="title" >
Chat Application </h1>
                <form
onSubmit={handleSubmit}>
                    <input type="text"
value={username} onChange={(e) =>
setUsername(e.target.value)}
className="input" required
placeholder="Username" />
```

```
                    <input type="password"
value={password} onChange={(e) =>
setPassword(e.target.value)}
className="input" required
placeholder="Password" />
                    <div align="center">
                        <button
type="submit" className="button">
                            <span>Start
Chatting</span>
                        </button>
                    </div>
                </form>
                <div
className="error">{error}</div>
            </div>
        </div>
    );
}

export default LoginForm
```

messageform.jsx

```
import { PictureOutlined, SendOutlined } from "@ant-design/icons";

import { useState } from "react"

import { sendMessage, isTyping } from "react-chat-engine";


const MessageForm = (props) => {


  const [value, setValue] = useState('');

  const { chatId, creds } = props;


  const handleSubmit = (event) => {

    event.preventDefault();
    const text = value.trim();

    if (text.length > 0) {

        sendMessage(creds, chatId, { text })

    }


    setValue('');

}


const handleChange = (event) => {

    setValue(event.target.value);


    isTyping(props, chatId);


}


const handleUpload = (event) => {

    sendMessage(creds, chatId, { files: event.target.files, text: '' })

}


return (

    <form className="message-form" onSubmit={handleSubmit}>

        <input

            placeholder="Type Your Message Here"

            className="input-message"

            value={value}

            onChange={handleChange}
            onSubmit={handleSubmit}
          />
          <label htmlFor="upload-button">
            <span className="image-button">
              <PictureOutlined className="picture-icon" />
            </span>
          </label>
          <input id="upload-button"
            type="file"
            multiple="false"
            style={{ display: 'none' }}
            onChange={handleUpload}
          />
          <button type="submit" className="send-button">
            <SendOutlined className="send-icon" />
          </button>
        </form>
    )
}

export default MessageForm;
```

mymessage.jsx

```
const MyMessage = ({ message }) => {
    if (message?.attachments?.length > 0) {
        return (
            <img
```

```
          src={message.attachments[0].file}
          alt="message-attachment"
          className="messgage-image"
          style={{ float: 'right' }}
        />
      )
    }
```

Theirmessages.jsx

```
const TheirMessage = ({ lastMessage, message }) => {


  const isFirstMessageByUser = !lastMessage || lastMessage.sender.username !==
message.sender.username;

  return (
    <div className="message-row">
      {isFirstMessageByUser && (
        <div className="message-avatar"
          style={{ backgroundImage: `url(${message?.sender?.avatar})` }} />
      )}


      {message?.attachments?.length > 0
        ? (
          <img
            src={message.attachments[0].file}
            alt="message-attachment"
            className="messgage-image"
            style={{ marginLeft: isFirstMessageByUser ? '4px' : '48px' }}
```

```
                />
              ) : (
                <div className="message" style={{ float: 'left', backgroundColor: '#CABCDC',
marginLeft: isFirstMessageByUser ? '4px' : '48px' }}>
                  {message.text}
                </div>
              )

          }
      </div>
   )
}

export default    console.log(message);
   return (
      <div className="message" style={{ float: 'right', marginRight: '18px', color: 'white',
backgroundColor: '#3B2A50' }}>
         {message.text}
      </div>
   )
}

export default MyMessage;
```

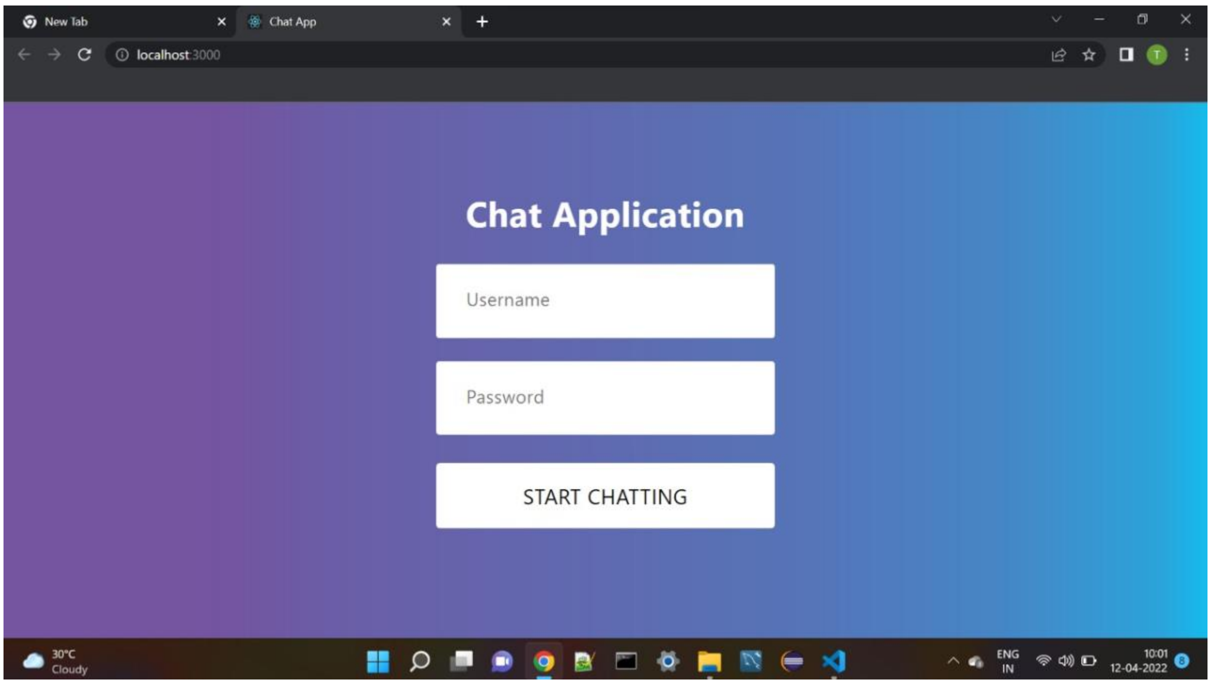
TheirMessage.jsx

```
const TheirMessage = ({ lastMessage, message }) => {
  const isFirstMessageByUser = !lastMessage || lastMessage.sender.username !==
message.sender.username;

  return (

    <div className="message-row">

      {isFirstMessageByUser && (

        <div className="message-avatar"

          style={{ backgroundImage: `url(${message?.sender?.avatar})` }} />

      )}


      {message?.attachments?.length > 0

        ? (

          <img

            src={message.attachments[0].file}

            alt="message-attachment"

            className="messgage-image"

            style={{ marginLeft: isFirstMessageByUser ? '4px' : '48px' }}

          />

        ) : (

          <div className="message" style={{ float: 'left', backgroundColor: '#CABCDC',
marginLeft: isFirstMessageByUser ? '4px' : '48px' }}>

            {message.text}

          </div>

        )


      }

    </div>

  )
```

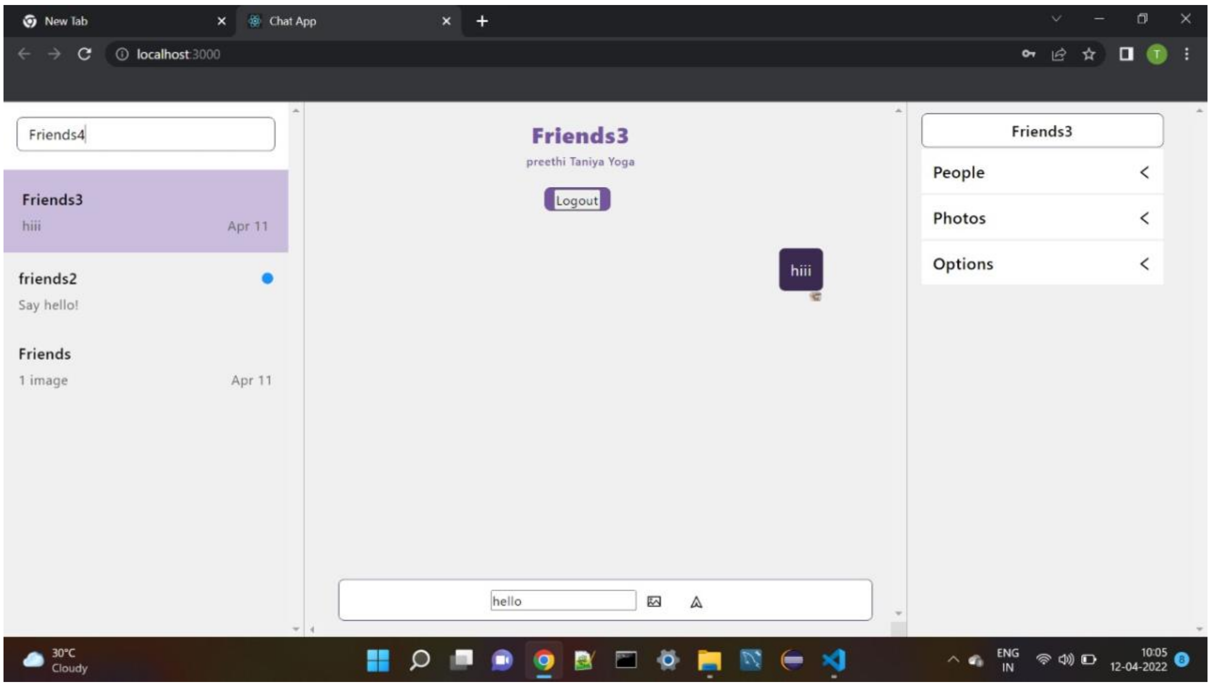
```
}

export default TheirMessage;
```

RESULT

login-page



HOME PAGE



New Tab

Chat App

localhost:3000

Friends4

Friends3
hiii Apr 11

friends2
Say hello!

Friends
1 image Apr 11

Friends3
preethi Taniya Yoga

Logout

hiii

hello

Friends3

People

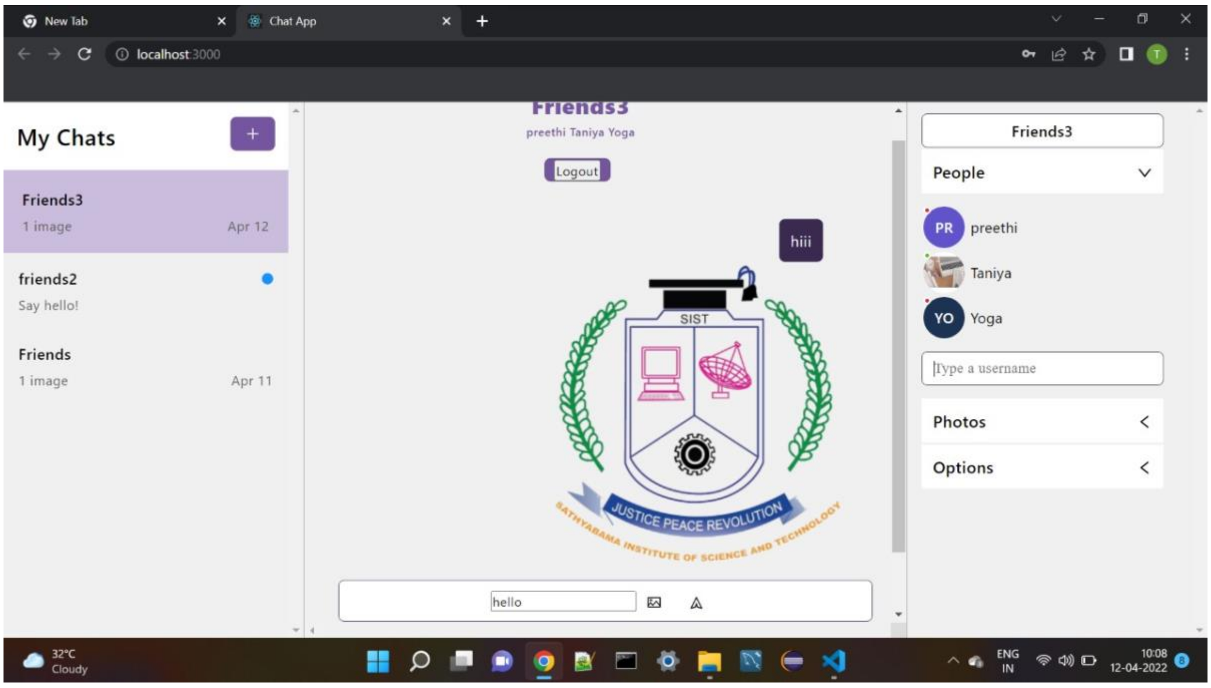
Photos

Options

30°C Cloudy

ENG IN 10:05 12-04-2022

SENDING MESSAGES



New Tab

Chat App

localhost:3000

My Chats

Friends3
1 image Apr 12

friends2
Say hello!

Friends
1 image Apr 11

Friends3
preethi Taniya Yoga

Logout

hiii

hello

Friends3

People

PR preethi

Taniya

YO Yoga

Type a username

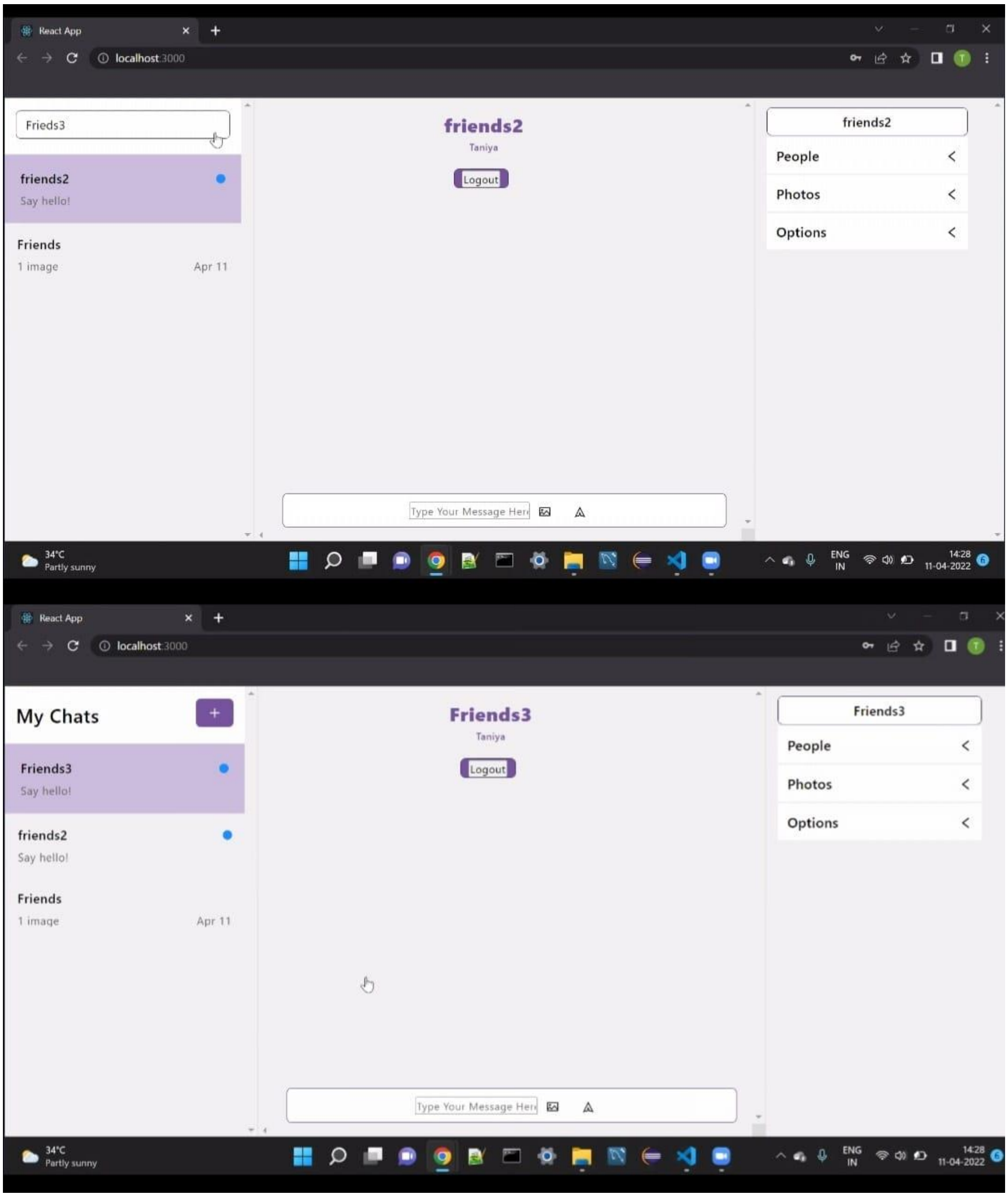
Photos

Options

32°C Cloudy

ENG IN 10:08 12-04-2022

CREATING A CHAT GROUP



CONCLUSION :

We have successfully created a Chat Apllicaiton using React js with the help of chat engine API key and hosted successfully on netlify