Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

B.Tech. CE Semester – V

Subject: (CE-515) Advanced Technologies

Project Title:

Online Chatting Application System

By:

Amethiya Yash G. (Roll No: CE002 ID: 18CEUBS026)

Guided by:

Prof. Ankit P. Vaishnav

Prof. Prashant M. Jadav

Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

CERTIFICATE

This is to certify that Advanced Technologies project entitled “Online Chatting System” is the bonafied report of work carried out by

Amethiya Yash G. (18CEUBS026)

Of Department of Computer Engineering, Semester V, academic year 2019-20, under our supervision and guidance.

Guide	Guide	HOD
Prof. Ankit P. Vaishnav Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad.	**Prof. Prashant M. Jadav** Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad.	**Dr. C. K. Bhensdadia** Head of the Department of Department of Computer Engineering, Dharmsinh Desai University, Nadiad.

Abstract

This web chat is a system that allows users to communicate in real-time using easily accessible web interfaces. It is a type of Internet online chat distinguished by its simplicity and accessibility to users who do not wish to take the time to install and learn to use specialized chat software. This trait allows users instantaneous access and only a web browser is required to chat. For any further query we have provided our email addresses and social media tags so that they can contact us any time. We thrive for user interaction and user satisfaction.

Introduction

This chat application is a web application which gives users a chance to communicate with any other user. The user login is authenticated. Users can login through their emails.

Users can see and select the person they are desired to chat with. When user joins any chat, the user is welcomed with message from the system and all the messages between two users are end to end encrypted. When User lefts the chat automatically his or her all chats are cleared automatically.

At last user can logout from the Chat application. If the user has any type of query they can also contact us through the email provided by us. They can also contact us through social media platforms like whatsapp, instagram, facebook. They can also call us for any type of query.

Technology Used:

- Cascading Style Sheet (CSS 3) for styling the HTML pages.
- JavaScript for providing dynamic content for the HTML pages.
- Jquery to provide dynamic data to the request pages.
- Bootstrap4 for styling the HTML pages using predefined classes of bootstrap.
- Angular for frontend purposes.
- Express.js for routing purposes.
- Node.js for backend purposes.
- Mongo DB as a database manager to add, update and fetch data from the database using Node.js mongoose module.
- Socket.io and Socket.io-client is a JavaScript library for real time web applications. It enables real time, bi-directional communication between web clients and servers. It has two parts: a client-side (Socket.io-client) library that runs in the browser, and a server-side (Socket.io) library for Node.

Platform Used:

- localhost/4200 for Angular
- localhost/4000 for Node js
- mongodb://127.0.0.1:27017
- localhost/3000 for socket.io

Tools Used:

- We used Github as a version control method and to manage the project.
- We used Visual Studio Code for development of the code.
- We used Postman for API functionality testing.

Software Requirement Specification

INDEX:

FUNCTIONAL REQUIREMENTS:

R1. MANAGE USER:

DESCRIPTION: This function allows user to create his/her account while he/she is using website for first time. This function also allows the user to login through his/her account and proceed further according to the needs. The user can also delete his/her account.

R1.1 Add User Info/Register User:

INPUT: User enters the First name, Last name, Date of birth, email, mobile number and password.
OUTPUT: Details will be forwarded to system and system will create the account.
PROCESS: The entered details will be authenticated internally and message is displayed accordingly.
NEXT: Navigated to Login page.

R1.2 User Login:

INPUT: The login page will be available for user, with his/her email address as username and Password he/she can Login.
OUTPUT: The home page will be displayed.
PROCESS: The email address and password will be authenticated and then output will be provided if successful.
NEXT: The home page will be shown.

R1.3 Delete User Account:

INPUT: User Selection.
OUTPUT: Confirmation Message is displayed.
PROCESS: The user account is removed from the database.
NEXT: Navigated to Login page.

R2. MANAGE FRIEND:

DESCRIPTION: We can add, remove, search friend using this function and user can block/report other user.

R2.1 Search Friend/User:

INPUT: User data like name, username or email address.
OUTPUT: List of all the user(s) matched or appropriate message is displayed.
PROCESS: The friend/user is checked internally and then displayed appropriately.

R2.2 Add Friend:

INPUT: User input on send request button or accept request button.
OUTPUT: Confirmation message.
PROCESS: Respective user(s) is/are to friend list.

R2.3 Remove Friend/User:

INPUT: User Selection.
OUTPUT: User is removed from friend list and Confirmation message is shown.

R2.4 Blocking/Reporting Friend/User:

INPUT: User Selection
OUTPUT: That particular user will be shifted to blocked list.

R3. MANAGE CHATTING:

DESCRIPTION: The list of all the chats between the two users are maintained in this function and user's conversation is automatically delete when he/she lefts the chat.

R3.1 Sending Messages:

INPUT: User types the message he/she wants to send and press send button or enter key.
OUTPUT: The message will be processed by the system and will be sent to the other user and The message sent will also be displayed to the user.

R3.2 Receiving Messages:

INPUT: No input.
OUTPUT: The message sent by the other user is displayed.

R3.3 Delete Messages:

INPUT: User Selection.
OUTPUT: The display page is shown.
PROCESS: All the messages or conversation are deleted.

R5. MANAGE CONTACT INFO:

DESCRIPTION: The user can connect with admin through this option.

R5.1 Contact Us:

INPUT: User Selection.
OUTPUT: Contact us page.

R5.2 Manage_Forget_Password:

INPUT: User Selection.
OUTPUT: An email will be sent to user to change his/her password.

NON FIONAL REQUIREMENTS:

1. Performance

The system shall give responses in a second after checking the input information. It should also be able to handle large number of requests or users at a time. Further uploading of data highly depends on the internet bandwidth and distance from the server.

2. Safety

The users' message and conversation are highly personal. The system has authorization to avoid any un-authorized access to users' messages or personal data.

SOFTWARE: Visual Studio Code, Github Desktop, MongoDB Compass

OS: Windows, Linux, ChromeOS

BROWSERS: Google Chrome, Microsoft Edge, Firefox.

Design Documents

1. XML:

User:

```
<users>
        <user id="id_01">
                <firstname>ABC</firstname>
                <lastname>XYZ</lastname>
                <dob>01-01-2001</dob>
                <email>ABC12345@gmail.com</email>
                <mobileno>9999999999</mobileno>
        </user>
</users>
```

2. DTD:

User:

```
<!DOCTYPE users[

<!ELEMENT users (user)*>

<!ATTLIST user id #REQUIRED>

<!ELEMENT user(firstname,lastname,dob,email,mobileno)>

<!ELEMENT firstname(#PCDATA)>

<!ELEMENT lastname(#PCDATA)>

<!ELEMENT dob(#PCDATA)>

<!ELEMENT email(#PCDATA)>

<!ELEMENT mobileno(#PCDATA)>

 ]>
```

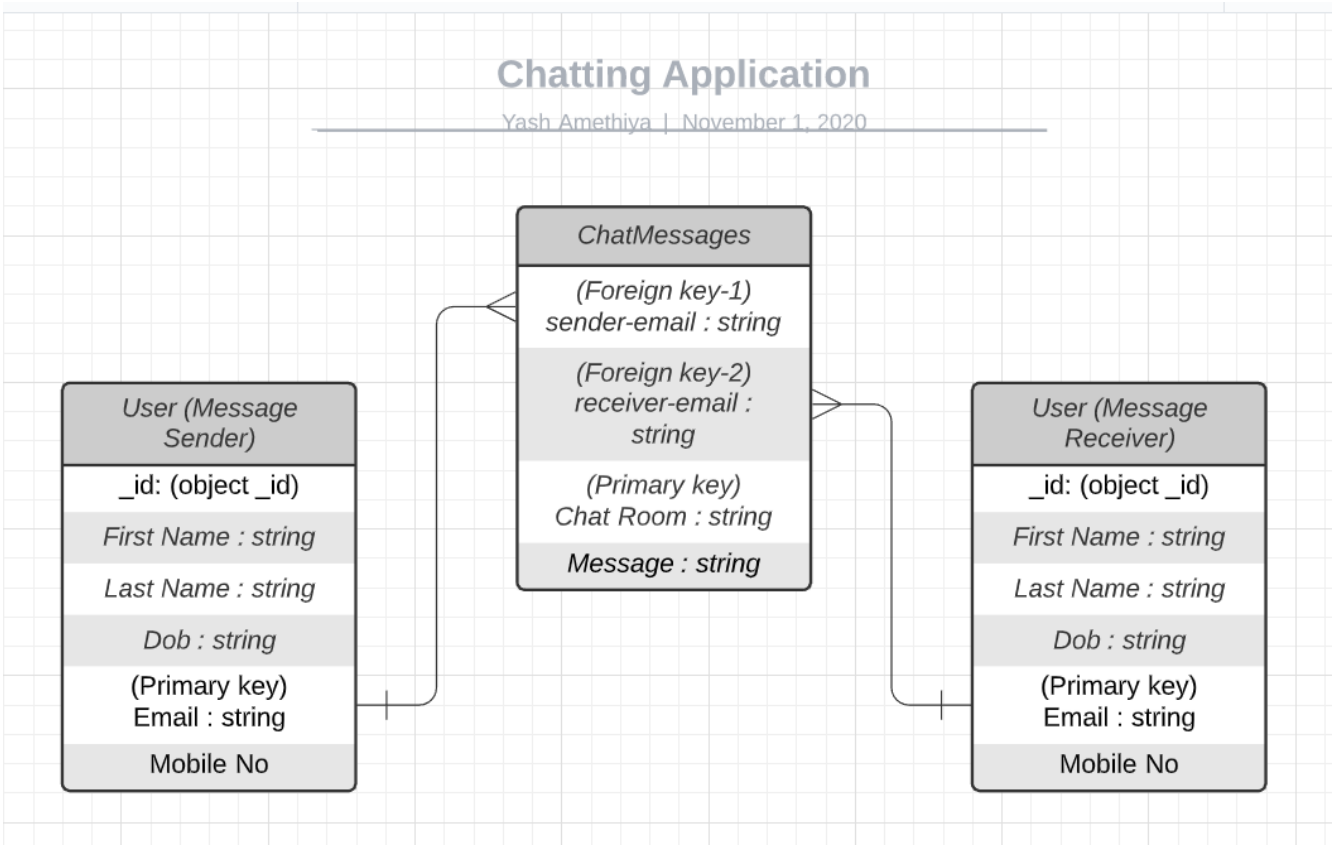
3. XSD:

User:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="users" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="users">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="user" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                        <xs:element name="firstname" minOccurs="0 maxOccurs="unbound">
                                                <xs:simpleType>
                                                        <xs:restriction base="xs:string">
                                                                <xs:maxLength value="30">
                                                        </xs:restriction>
                                                </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="lastname" minOccurs="0 maxOccurs="unbound">
                                                <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:maxLength value="30">
                                                </xs:restriction>
                                                </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="dob" minOccurs="0" maxOccurs="unbound">
                                                <xs:simpleType>
                                                        <xs:restriction base="xs:string">
                                                                <xs:maxLength value="10">
                                                        </xs:restriction>
                                                </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="email" minOccurs="0" maxOccurs="1">
                                                <xs:simpleType>
                                                        <xs:restriction base="xs:string">
                                                                <xs:maxLength value="60">
```

```
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                            <xs:element name="mobileno" minOccurs="0" maxOccurs="unbound">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:unsignedLong">
                                            <xs:maxLength value="10">
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                            </xs:sequence>
                            <xs:attribute name="id" type="xs:string" use="optional"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

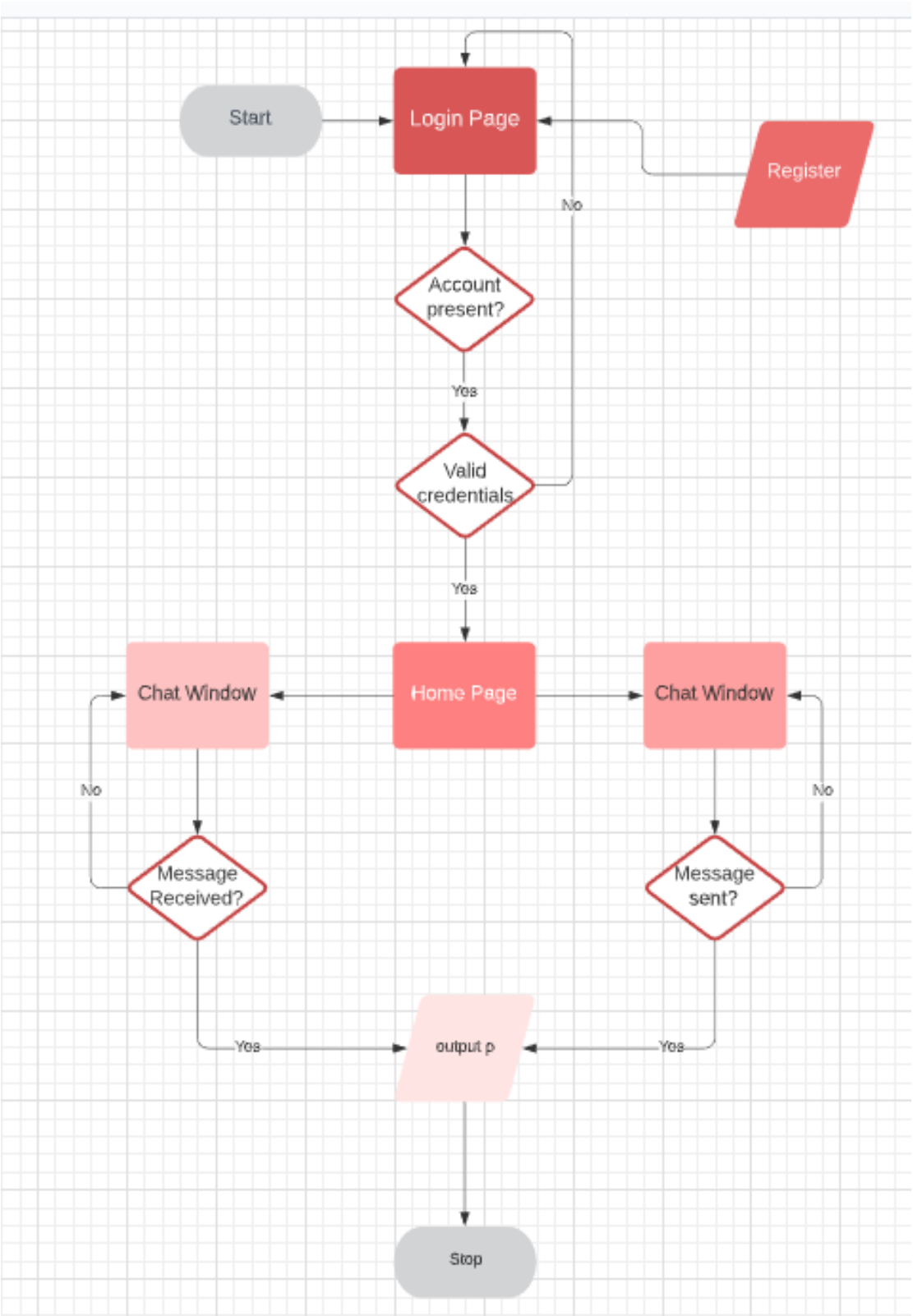
4. E-R Diagram:


Chatting Application
Yash Amethiya | November 1, 2020
ChatMessages
(Foreign key-1)
sender-email : string
(Foreign key-2)
receiver-email :
string
(Primary key)
Chat Room : string
Message : string
User (Message
Sender)
_id: (object _id)
First Name : string
Last Name : string
Dob : string
(Primary key)
Email : string
Mobile No
User (Message
Receiver)
_id: (object _id)
First Name : string
Last Name : string
Dob : string
(Primary key)
Email : string
Mobile No

5. Data Dictionary:

User								
Sr. No.	**Filed Name**	**Data Type**	**Width**	**Required**	**Unique**	**PK/FK**	**Referenced Table**	**Description**
1	_id	object	200	Yes	Yes			AUTO INCREMENT
2	firstname	varchar	30	Yes	No			
3	lastname	varchar	30	Yes	No			
4	dob	varchar	10	Yes	No			
5	email	varchar	60	Yes	Yes	PK		
6	mobileno	integer	10	No	No			

6. Flow Chart:


Start
Login Page
Register
No
Account present?
Yes
Valid credentials
Yes
Chat Window
Home Page
Chat Window
No
No
Message Received?
Message sent?
Yes
output p
Yes
Stop

Implementation Details

Login and Register Module:

We used auth guard service for authentication and authorization and to maintain the session for the whole project by generating a unique token and saving it to the users' local storage. User can add his/her other information and register/login his/herself in.

Chat Module:

After coming onto the home page user would be able to see list of all user with whom he/she can chat with or message to. After clicking on any user the chat window of that particular user is popped up onto the right side of the page where user can send to and will receive message from the opposite user. After close the chat window or moving onto the other chat the previous messages or conversation with the user will be deleted automatically.

Logout Module:

Whenever user wants to logout he/she can find a logout button a menu dropdown button and can left the application. When user clicks on the logout button he/she gets redirect to login page and get logged out and the token saved in local storage also gets deleted and so user session gets over and a shows successful message.

Delete Module:

Whenever user wants to delete his/her account he/she can find a delete account button a menu dropdown button and can permanently delete his account from the application. When user clicks on the delete button he/she gets redirect to login page and account gets deleted and the token saved in local storage also gets deleted and so user session gets over and a shows successful messages for account getting deleted.

Admin Module:

We use mongodb compass as admin to manage users. We get all the statistics and data for all the users which have signed up to our application.

Function Prototype:

Database Schemas for MongoDB for User data:

```
var mongoose = require('mongoose')
var bcrypt = require('bcrypt');
var userSchema = mongoose.Schema({
    firstname: { type: String, require: true },
    lastname: { type: String, require: true },
    dob: { type: String, require: true },
    email: { type: String, require: true },
    mobileno: { type: Number, require: true },
    pswd: { type: String, require: true }
}, { timestamp: true })

userSchema.statics.hashPassword = function hashPassword(pswd) {
    return bcrypt.hashSync(pswd, 10);
}

userSchema.methods.isValid = function(hashedpassword) {
    return bcrypt.compareSync(hashedpassword, this.pswd);
}

module.exports = mongoose.model('user', userSchema)
```

Sending and Receiving Messages using socket.io:

```
var io = require('socket.io')(3000);
io.on('connection', socket => {
    console.log("new connection...")

    socket.on('join', function(data) {
        socket.join(data.roomName);
        console.log(data.firstname + ', you are Online chatting with ' + data.roomName);
        socket.emit('connected to chat', {
            firstname: data.firstname,
            message: 'Chatbot here!! You are connected with ' + data.chatName + '.'
        })
    })

    socket.on('message', function(data) {
        io.in(data.roomName).emit('new message', { firstname: data.firstname, message: data.message })
    })
});
```

Getting All User data:

```
router.post('/readUser', function(req, res, next) {
    User.find({}, (err, user) => {
        if (err) {
            console.log(err);
            res.json({
                msg: "Error"
            })
        } else {
            res.json({ user: user })
        }
    })
});
```

Token Verification

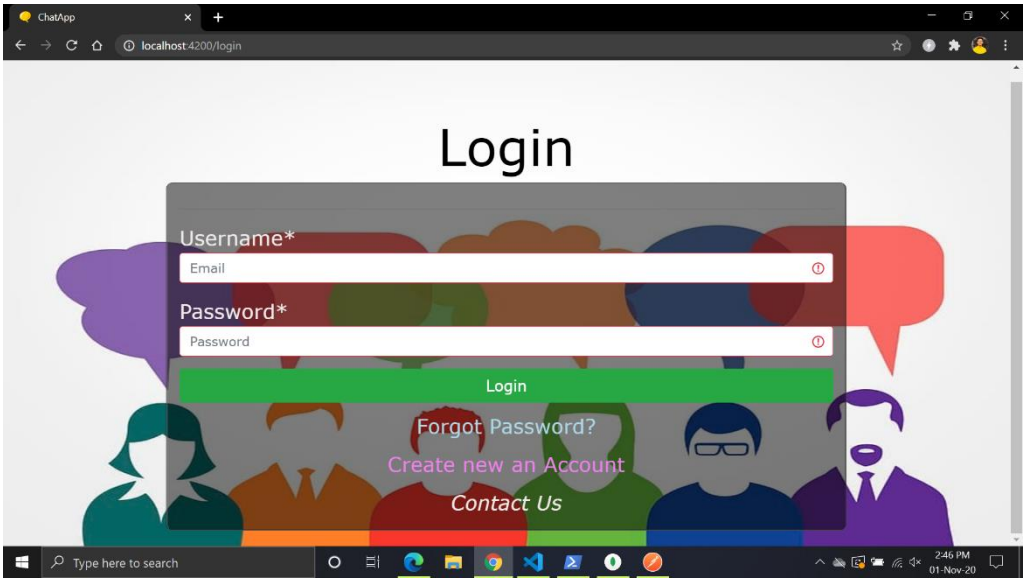
```
function verifyToken(req, res, next) {
    if (!req.headers.authorization) {
        return res.status(401).send('Unauthorized request')
    }
    let token = req.headers.authorization.split(' ')[1]
    if (token === 'null') {
        return res.status(401).send('Unauthorized request')
    }
    let payload = jwt.verify(token, 'secretkey@2001')
    if (!payload) {
        return res.status(401).send('Unauthorized request')
    }
    req.userId = payload.subject
    next()
}
```

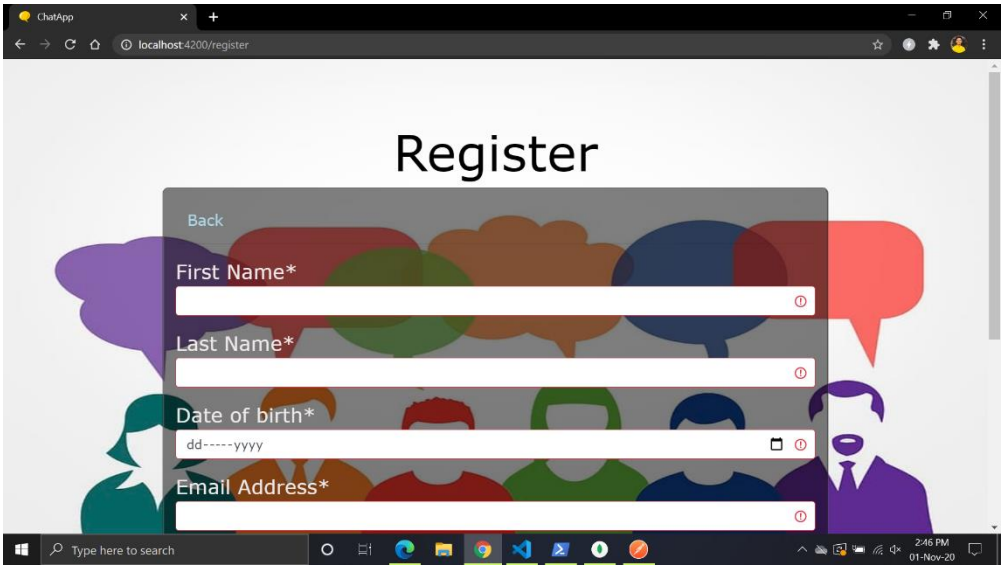
Testing

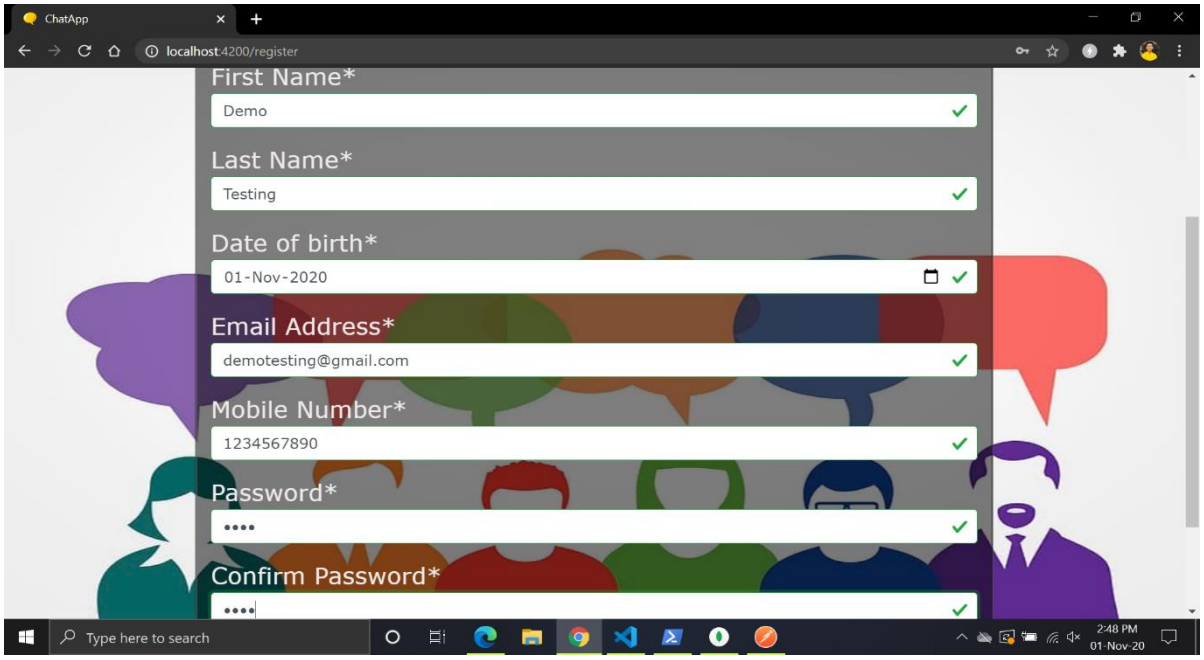
Testing Methods:

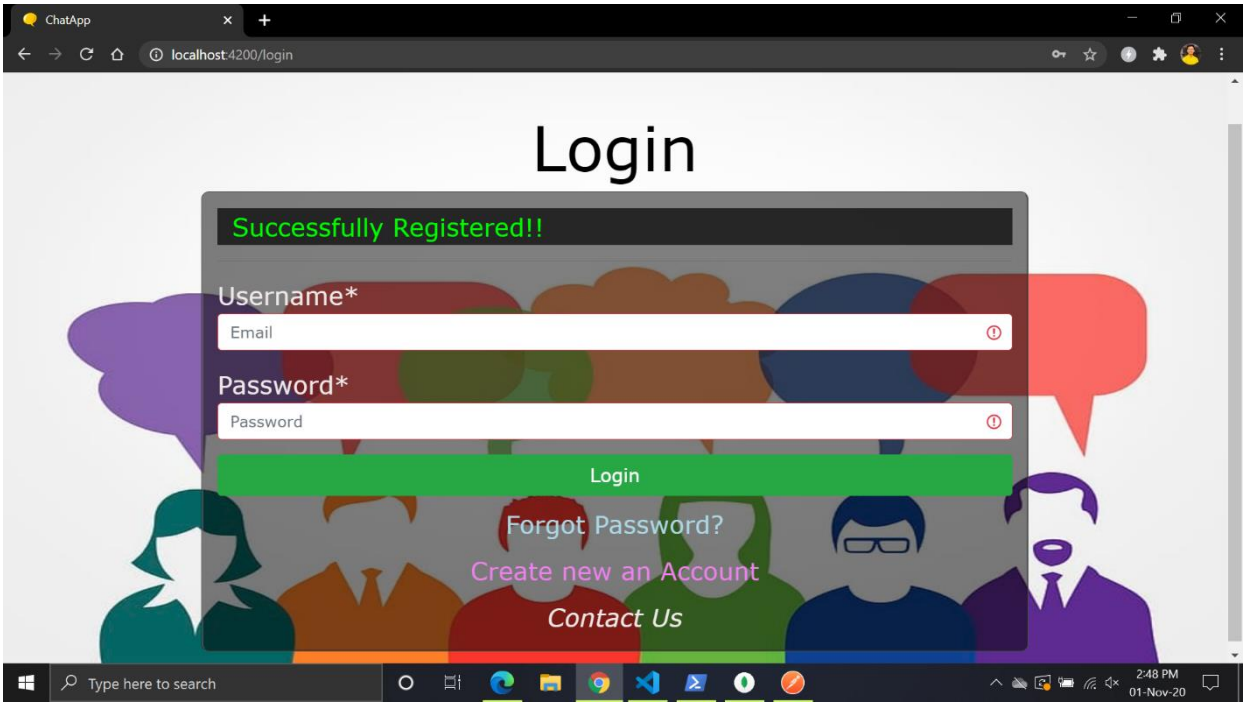
- We have used postman for testing node server functionalities.
- Form and input validations are done using Angular FormValidation

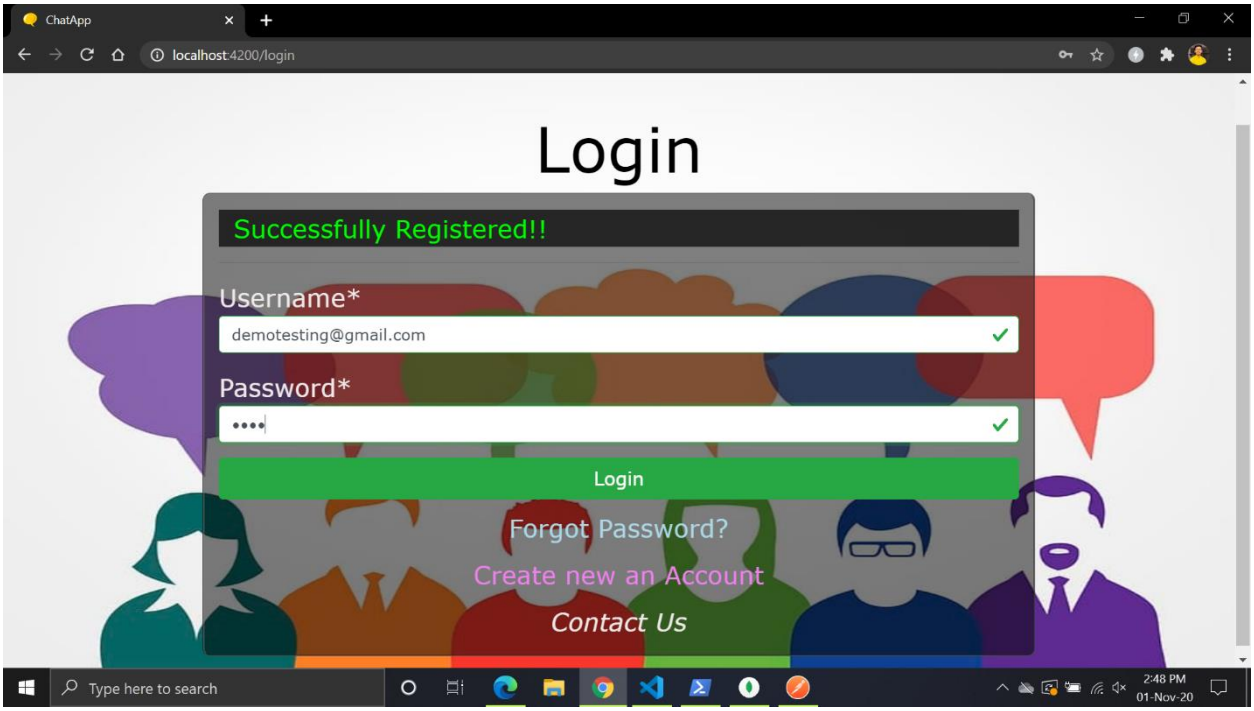
Screen-Shots

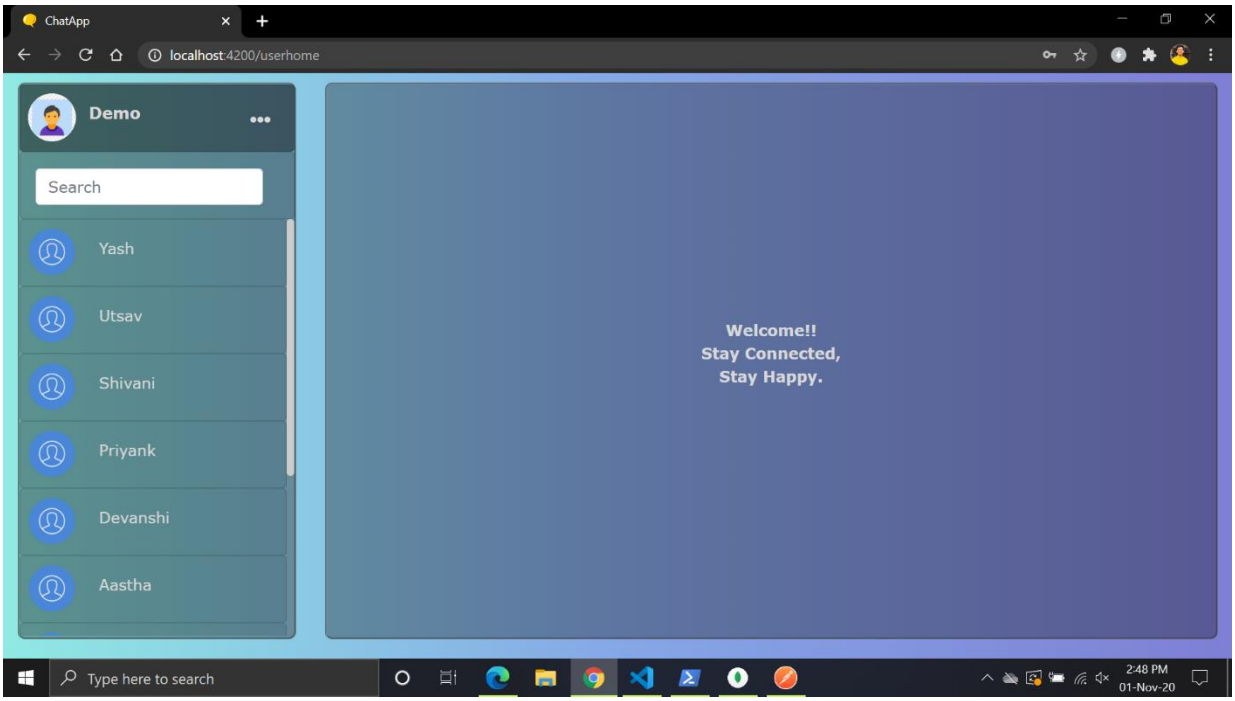




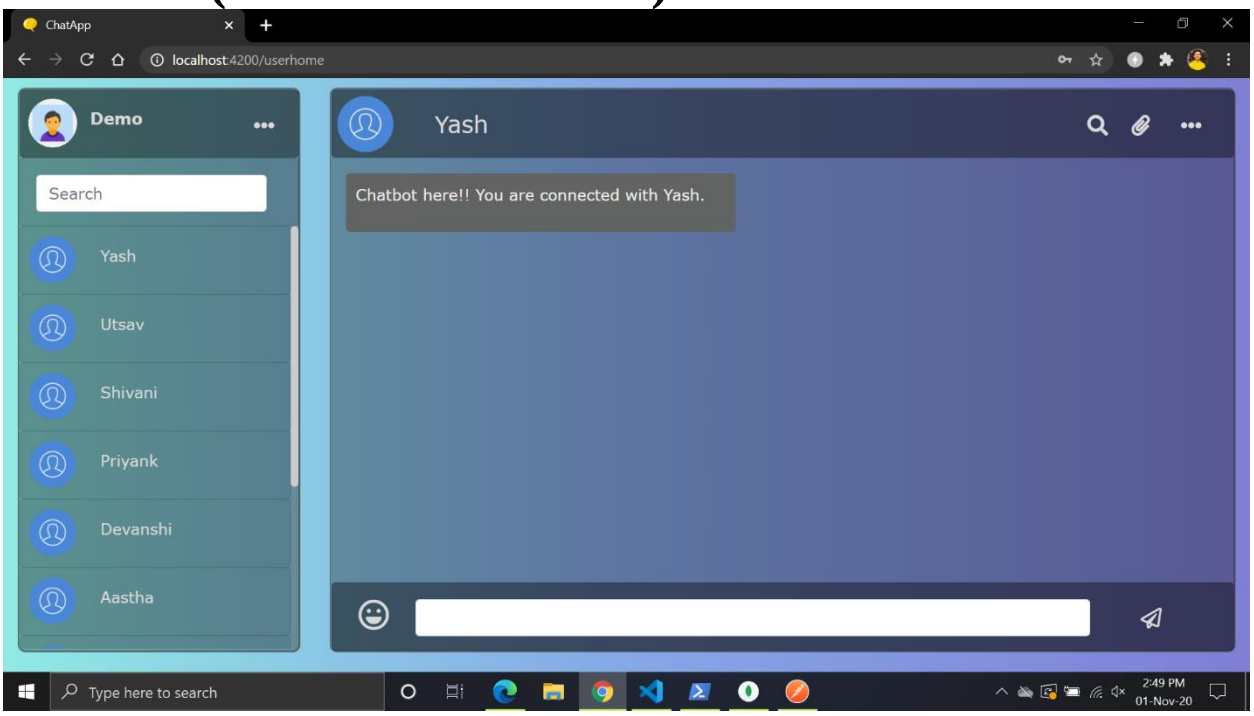

ChatApp
localhost:4200/register
First Name*
Demo
Last Name*
Testing
Date of birth*
01-Nov-2020
Email Address*
demotesting@gmail.com
Mobile Number*
1234567890
Password*
Confirm Password*
Type here to search
2:48 PM
01-Nov-20


ChatApp
localhost:4200/login
Login
Successfully Registered!!
Username*
Email
Password*
Password
Login
Forgot Password?
Create new an Account
Contact Us
Type here to search
2:48 PM
01-Nov-20

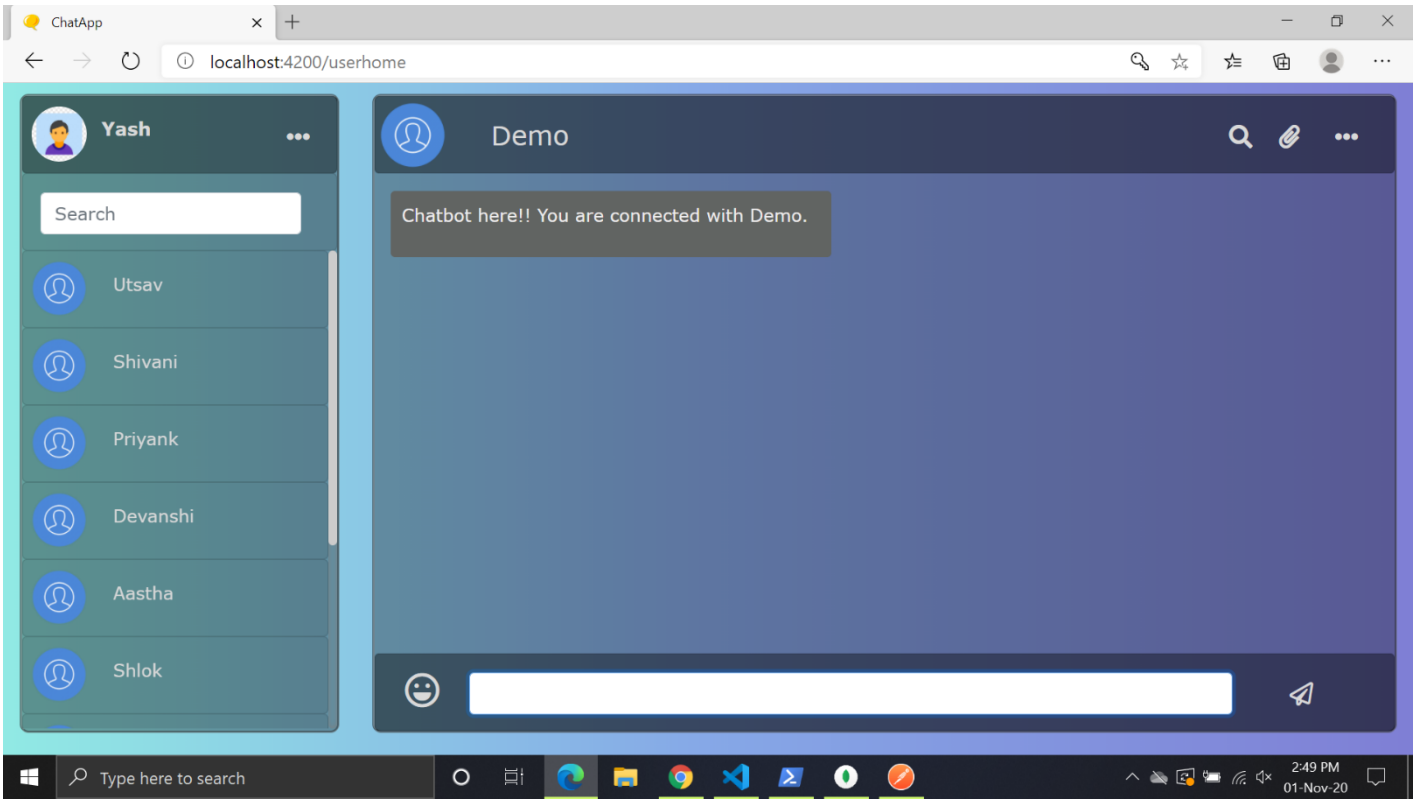

ChatApp
localhost:4200/login
Login
Successfully Registered!!
Username*
demotesting@gmail.com
Password*
Login
Forgot Password?
Create new an Account
Contact Us
Type here to search
2:48 PM
01-Nov-20

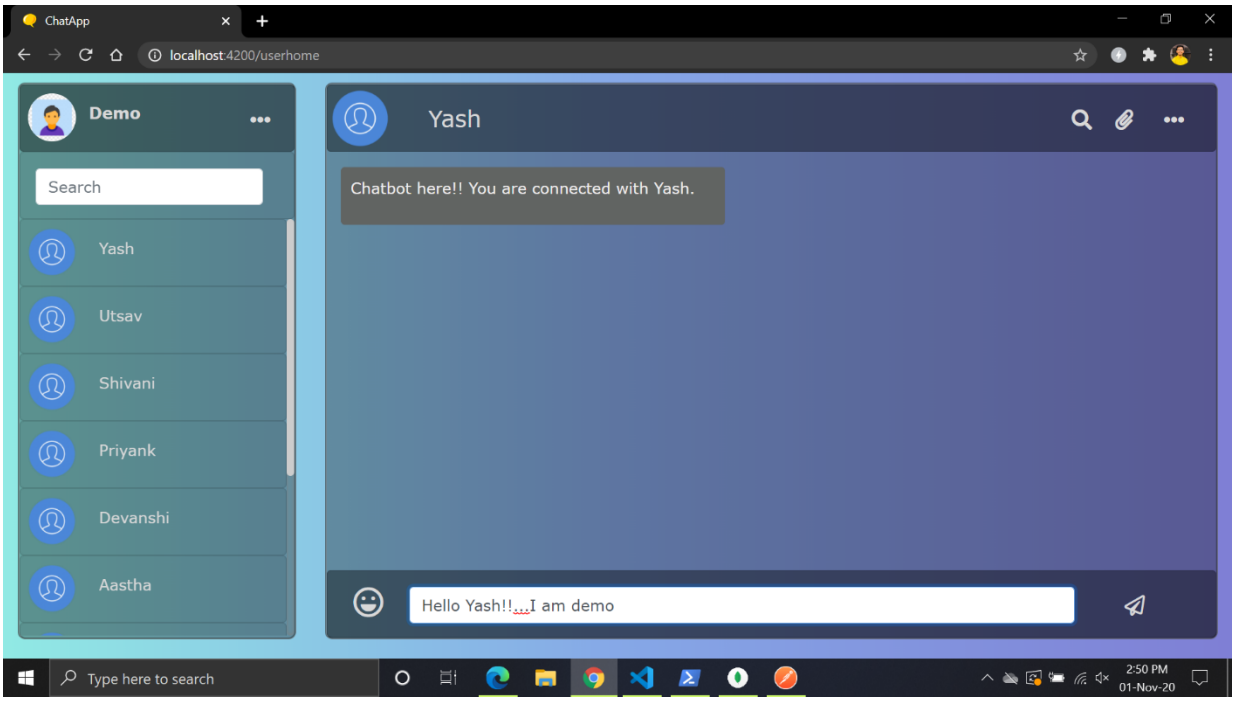

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Welcome!!
Stay Connected,
Stay Happy.
Type here to search
2:48 PM
01-Nov-20

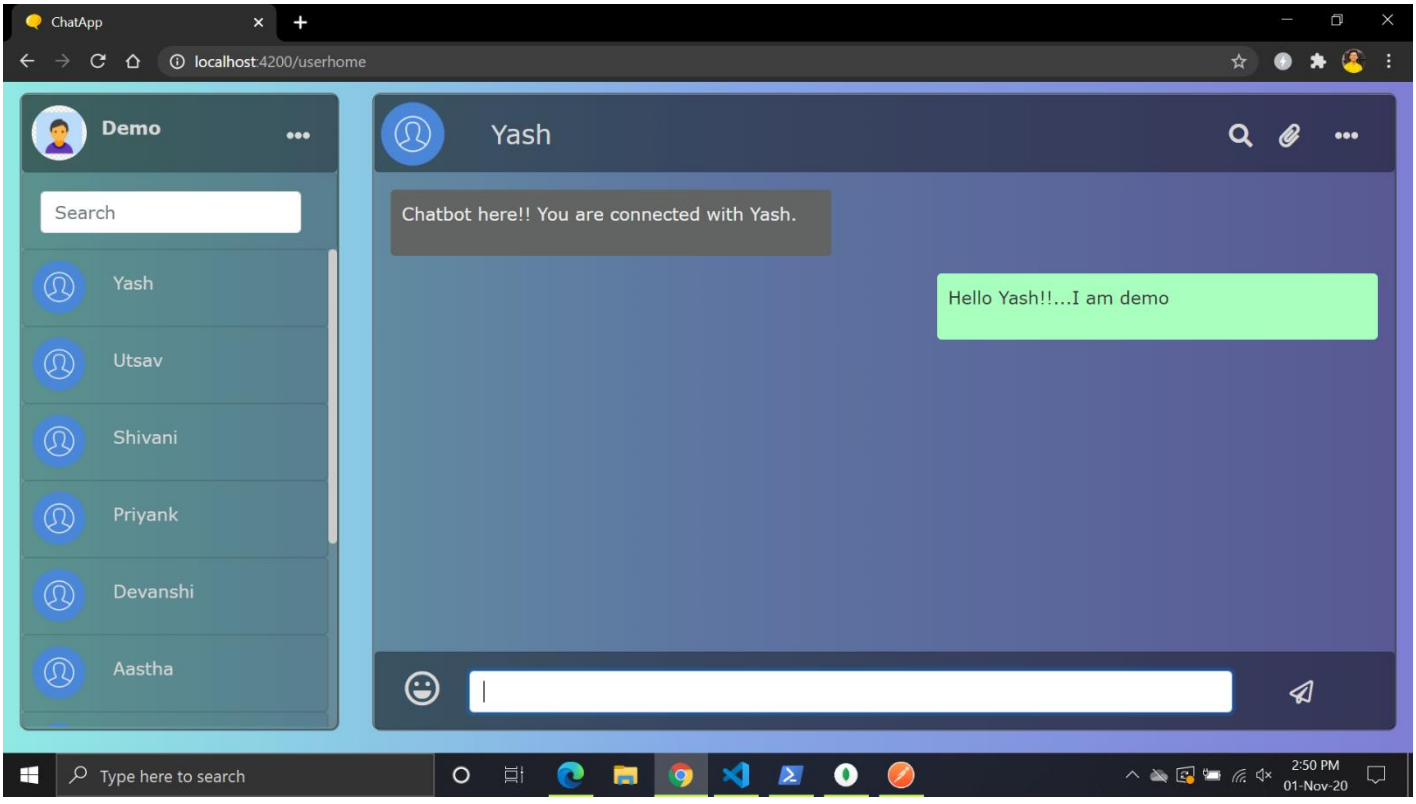
Demo (Demo is in chrome) is online with Yash

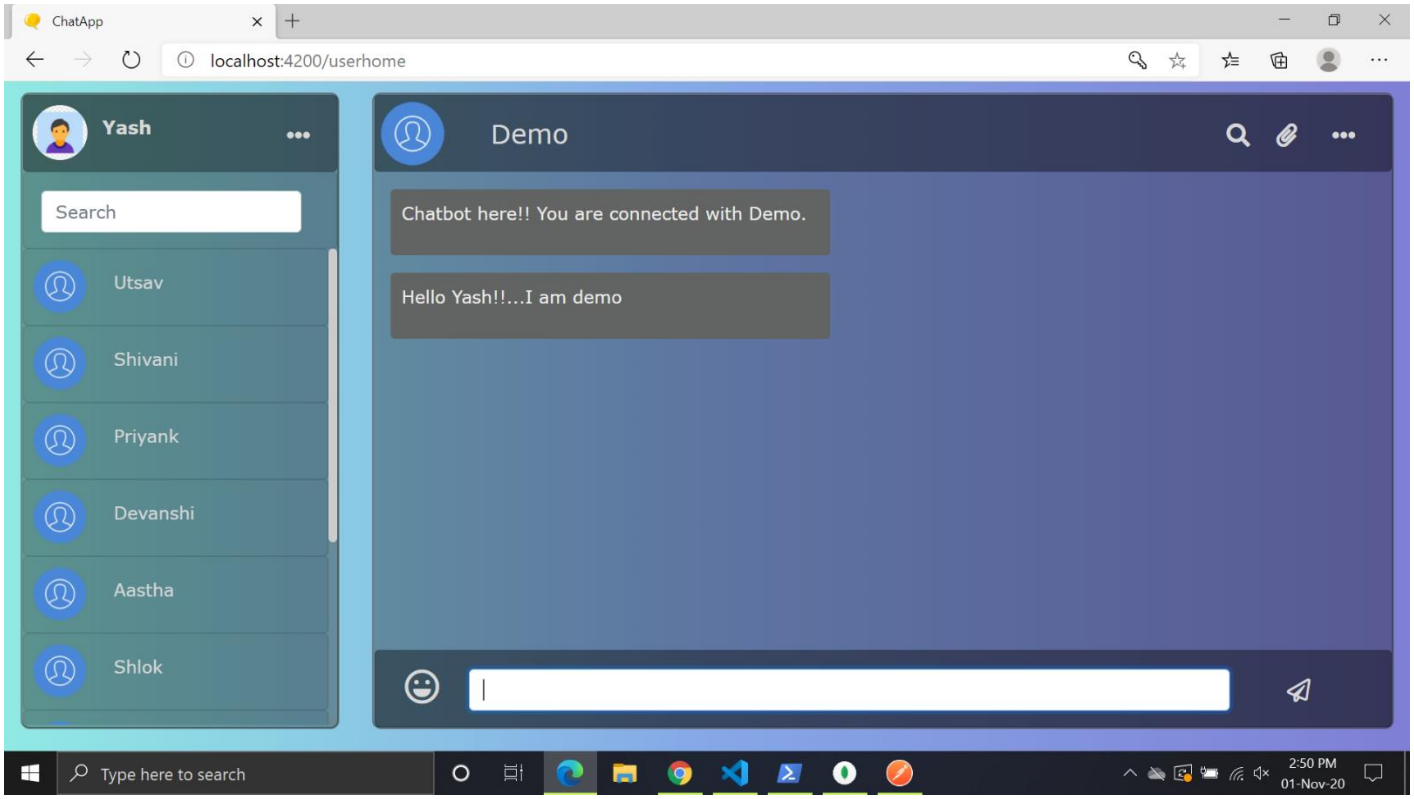


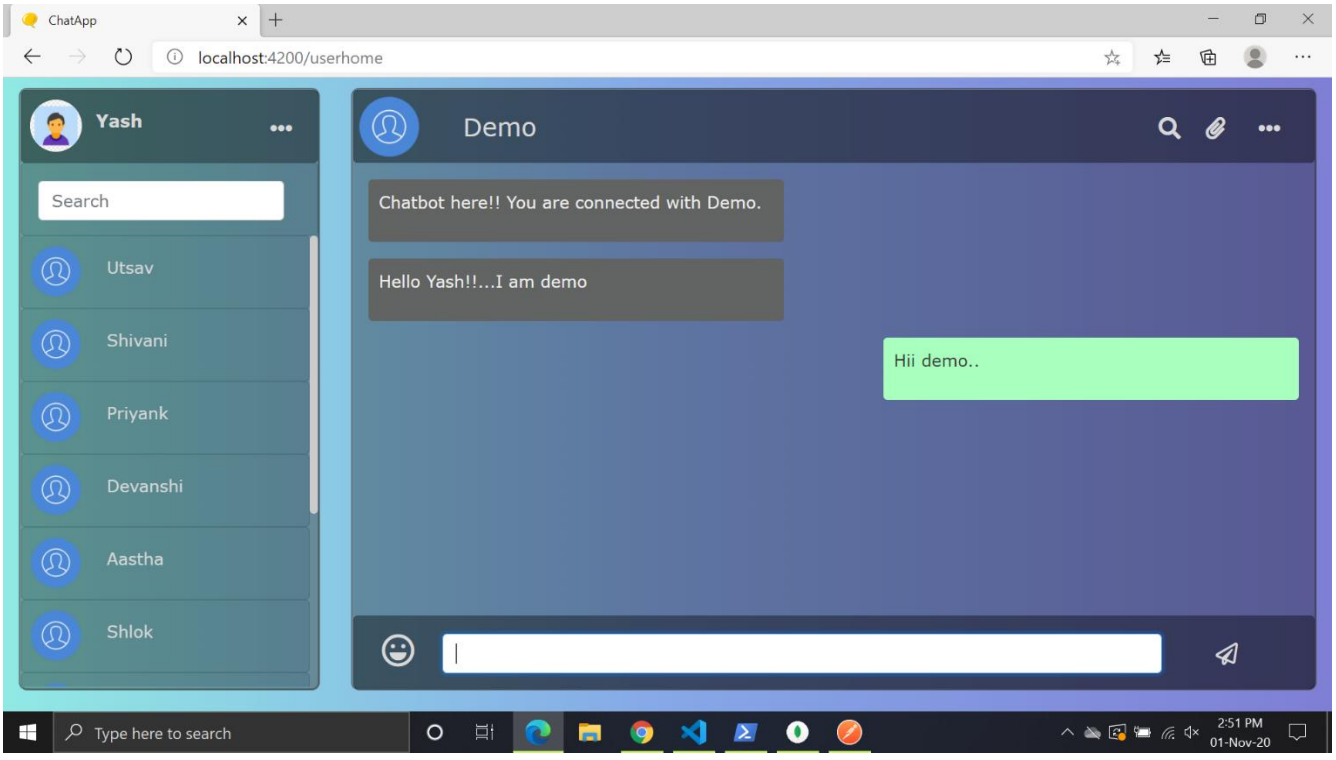
Yash (Yash is in Microsoft Edge) is also online with Demo

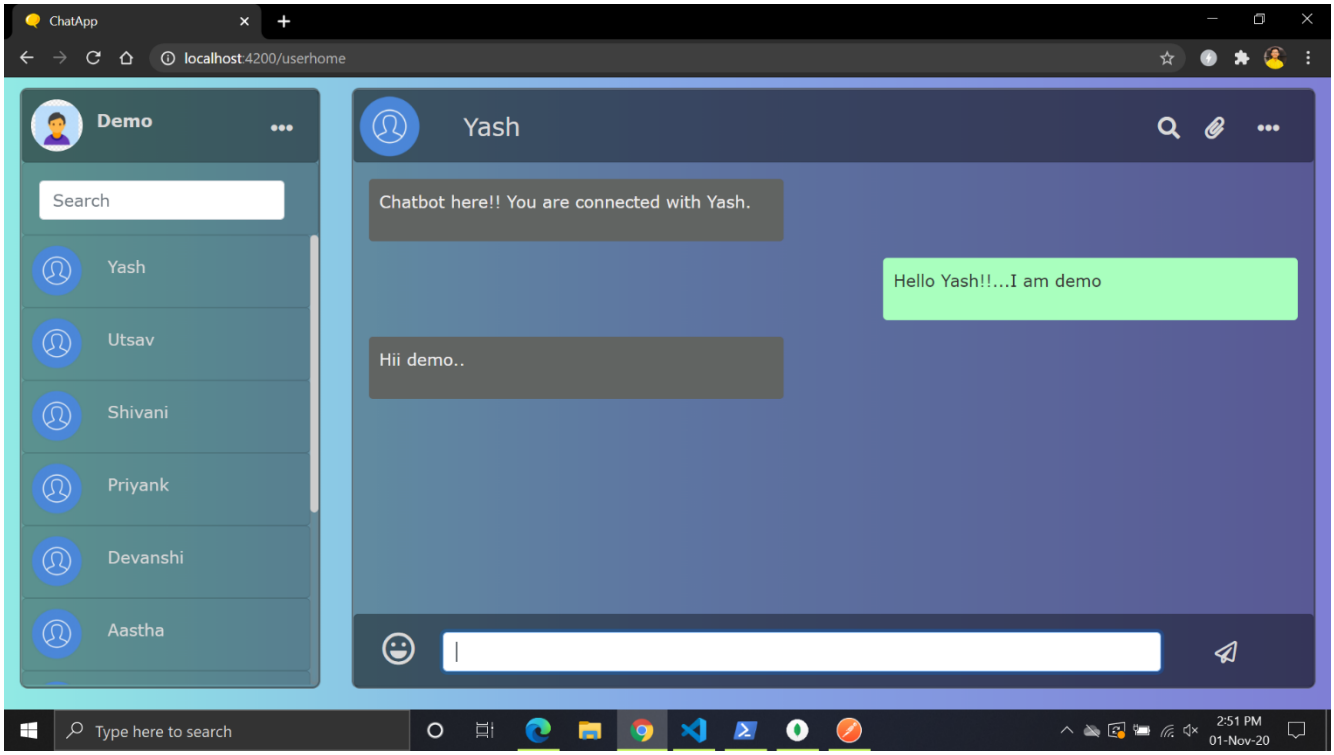


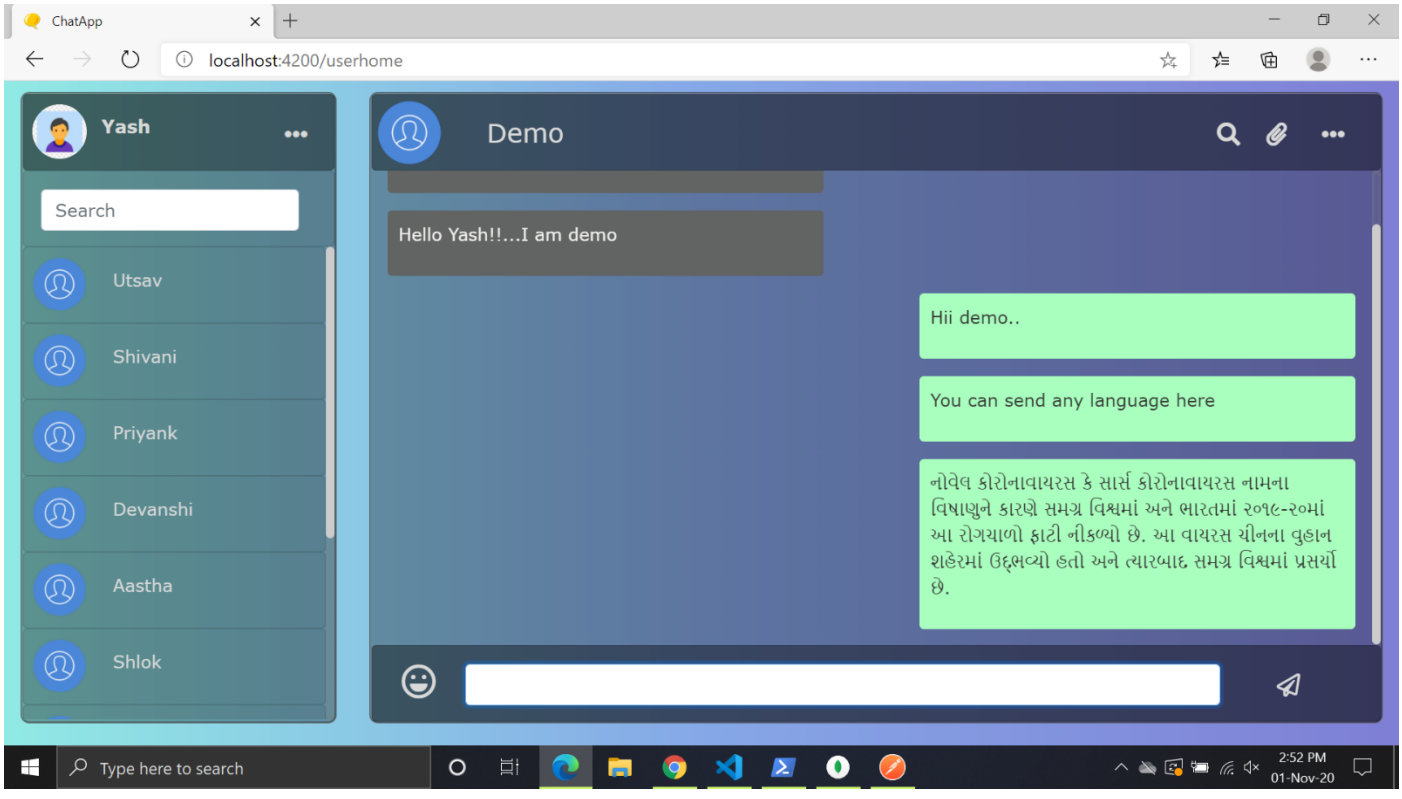

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
Chatbot here!! You are connected with Yash.
Hello Yash!!...I am demo
Type here to search
2:50 PM
01-Nov-20

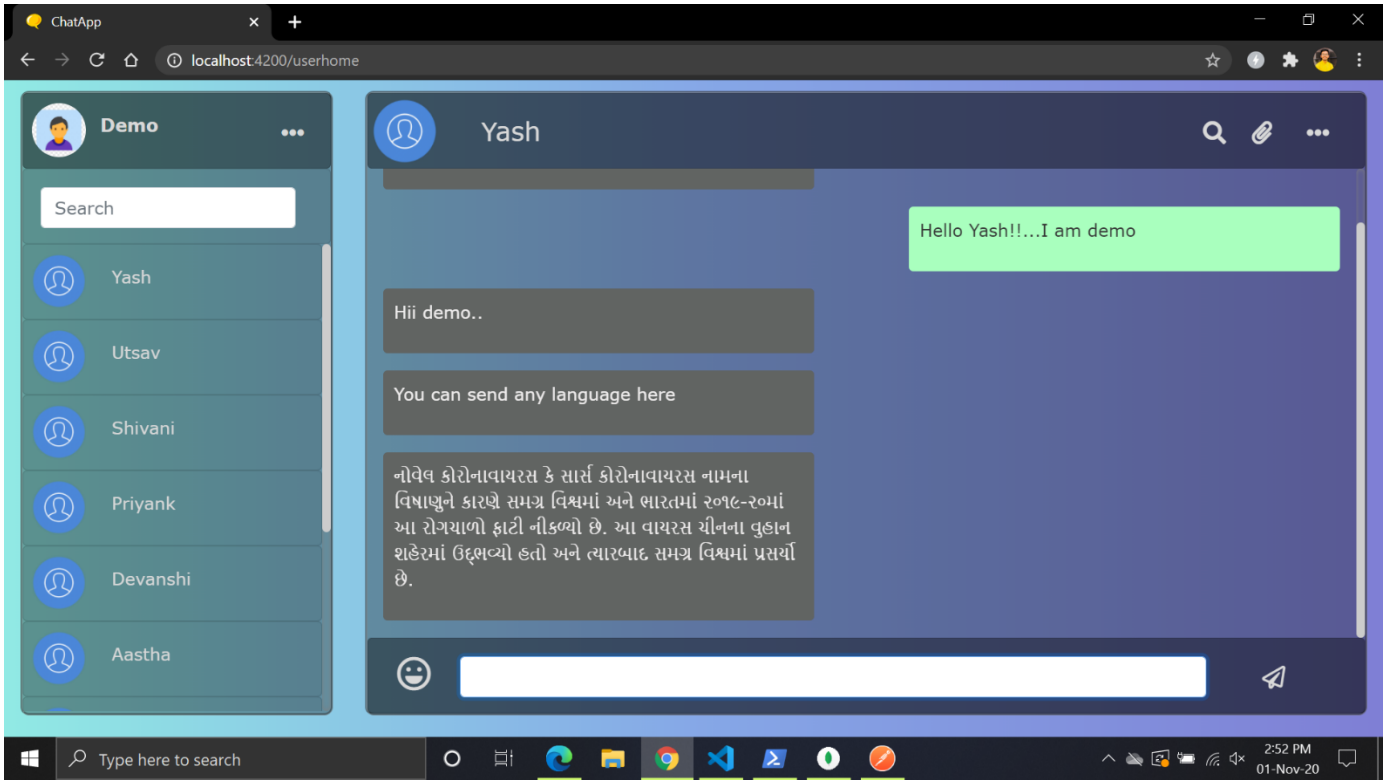

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
Chatbot here!! You are connected with Yash.
Hello Yash!!...I am demo
Type here to search
2:50 PM
01-Nov-20

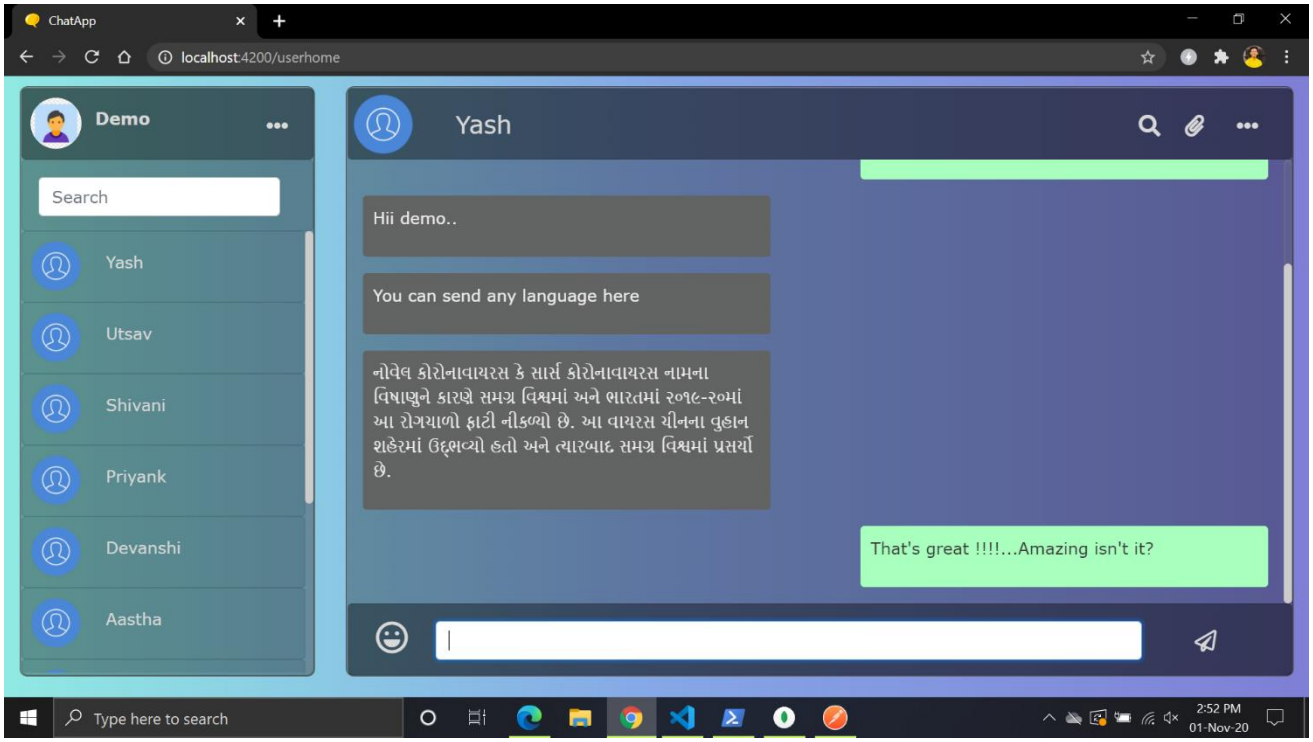

ChatApp
localhost:4200/userhome
Yash
Search
Utsav
Shivani
Priyank
Devanshi
Aastha
Shlok
Demo
Chatbot here!! You are connected with Demo.
Hello Yash!!...I am demo
Type here to search
2:50 PM
01-Nov-20

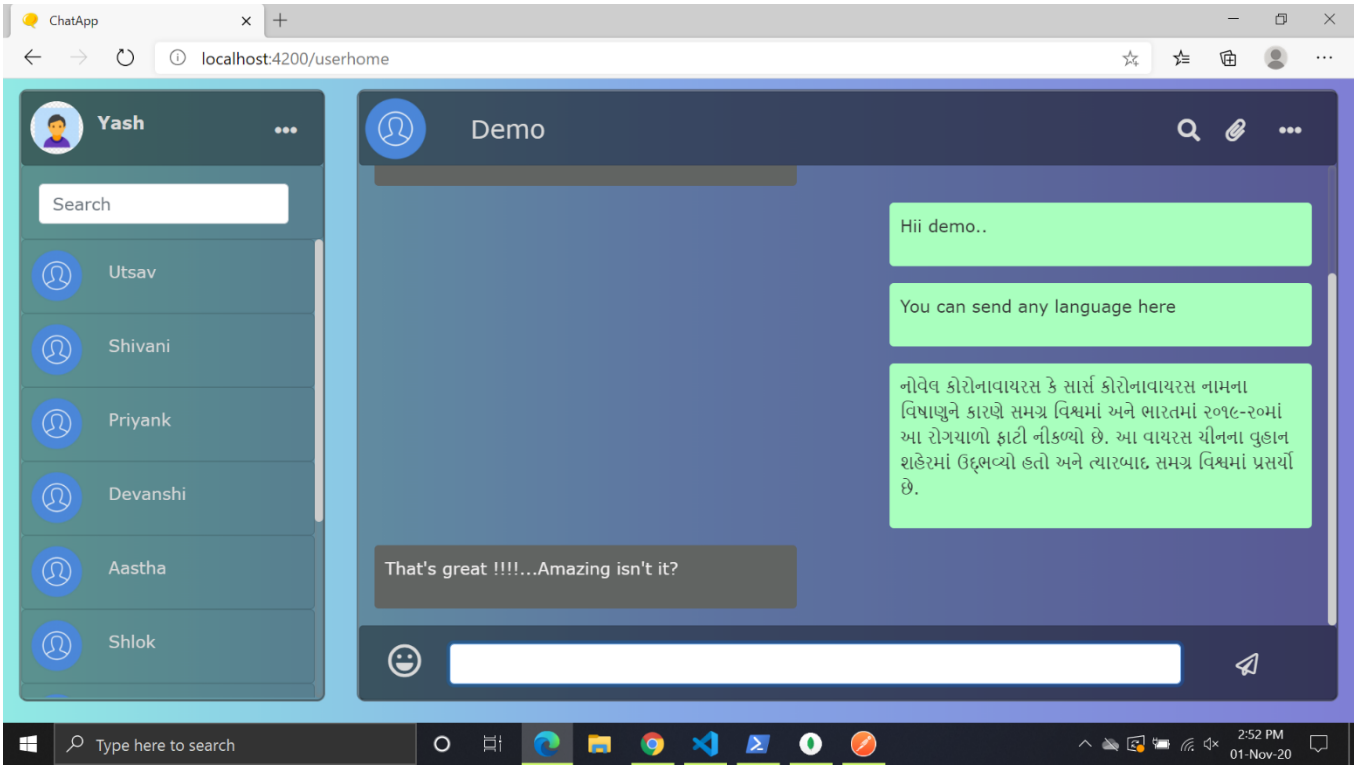

ChatApp
localhost:4200/userhome
Yash
Search
Utsav
Shivani
Priyank
Devanshi
Aastha
Shlok
Demo
Chatbot here!! You are connected with Demo.
Hello Yash!!...I am demo
Hii demo..
Type here to search
2:51 PM
01-Nov-20

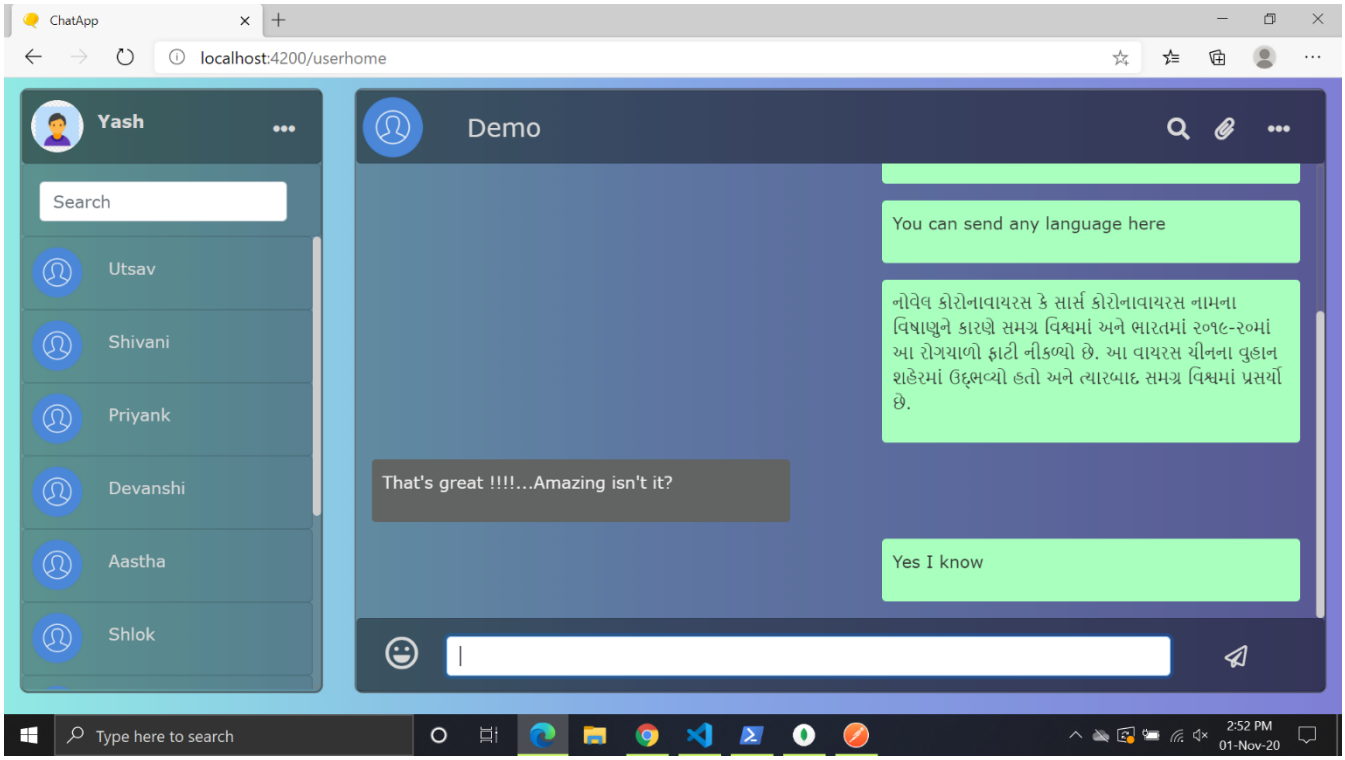

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
Chatbot here!! You are connected with Yash.
Hello Yash!!...I am demo
Hii demo..
Type here to search
2:51 PM
01-Nov-20

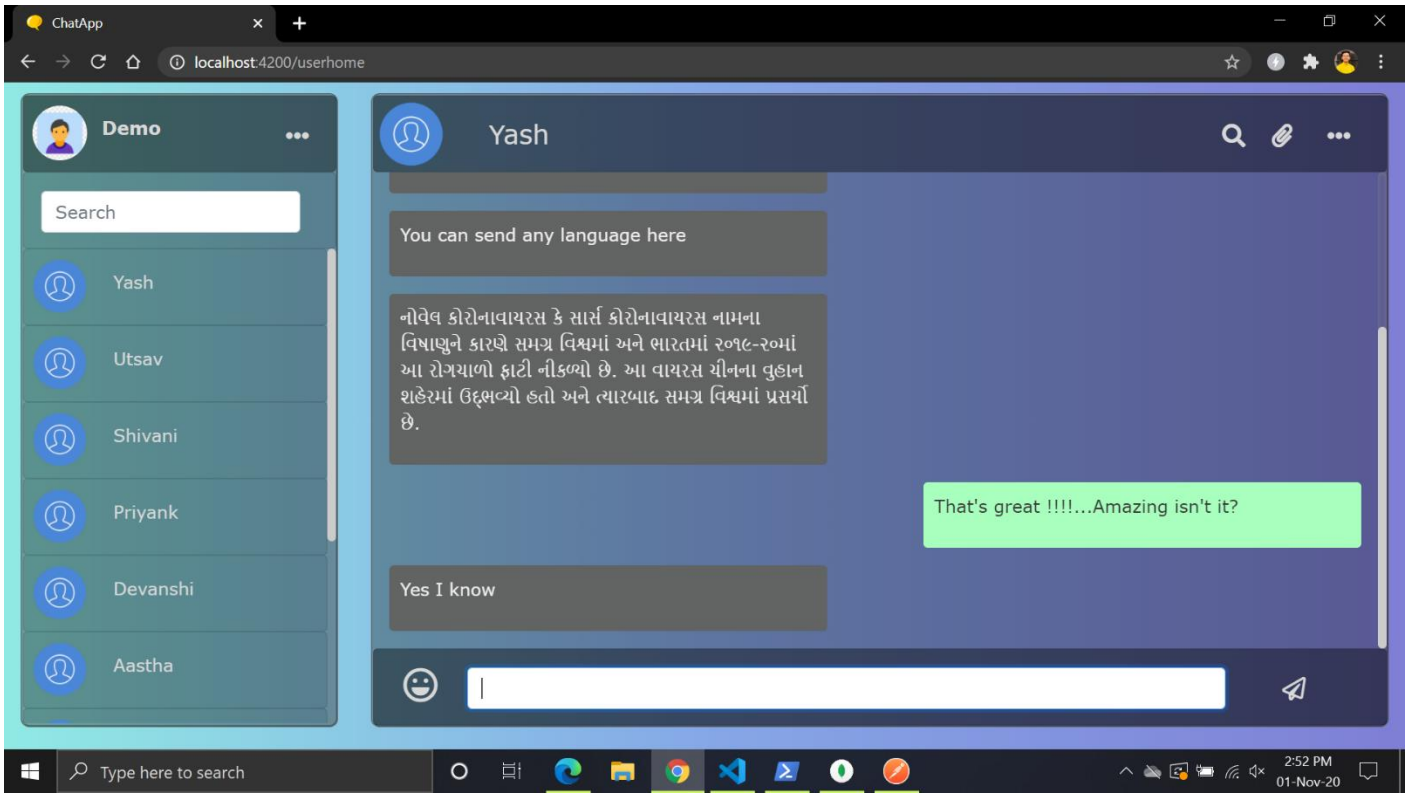

ChatApp
localhost:4200/userhome
Yash
Search
Utsav
Shivani
Priyank
Devanshi
Aastha
Shlok
Demo
Hello Yash!!...I am demo
Hii demo..
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્‌ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
Type here to search
2:52 PM
01-Nov-20

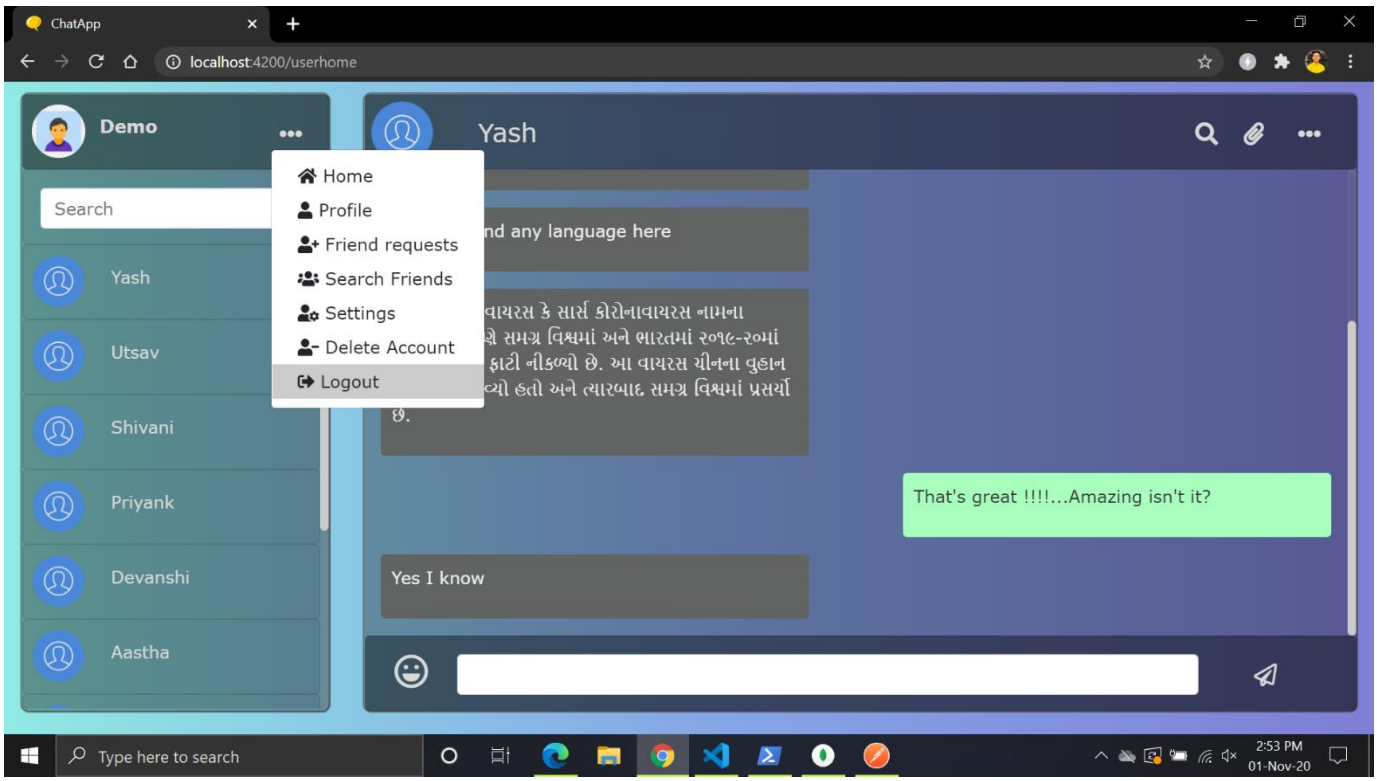

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
Hello Yash!!...I am demo
Hii demo..
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
Type here to search
2:52 PM
01-Nov-20

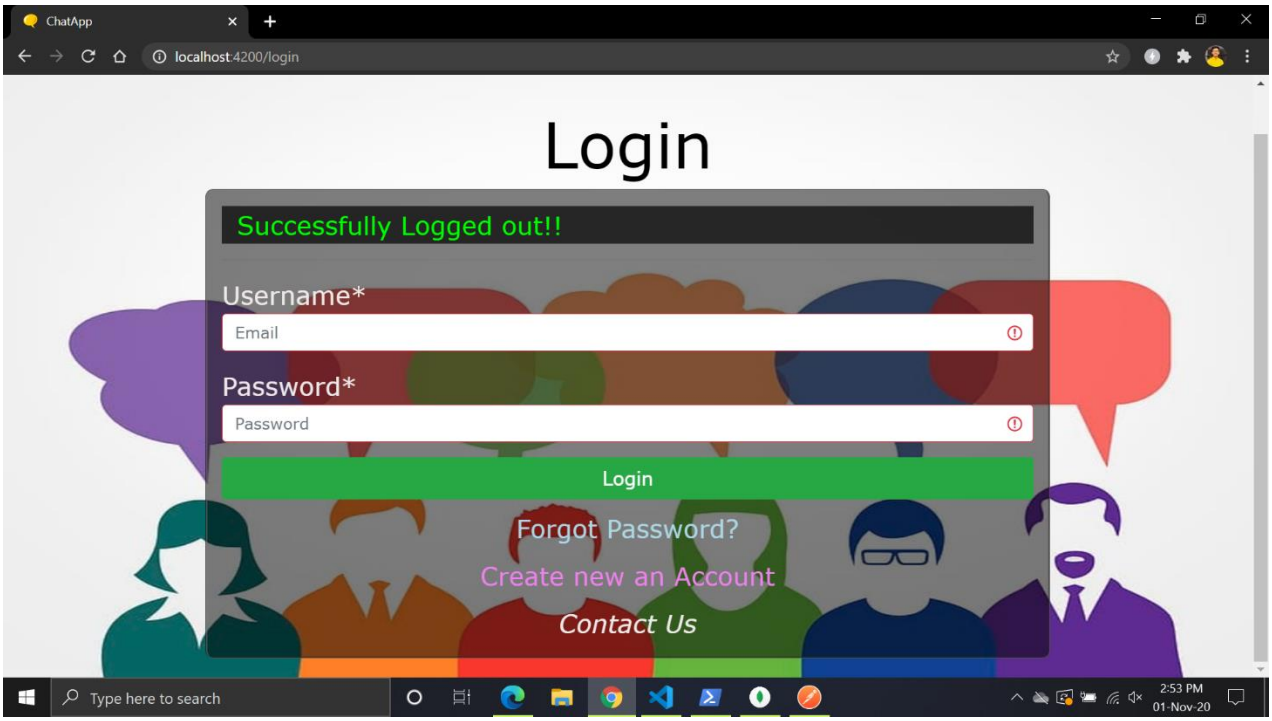

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
Hii demo..
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
That's great !!!!...Amazing isn't it?
Type here to search
2:52 PM
01-Nov-20

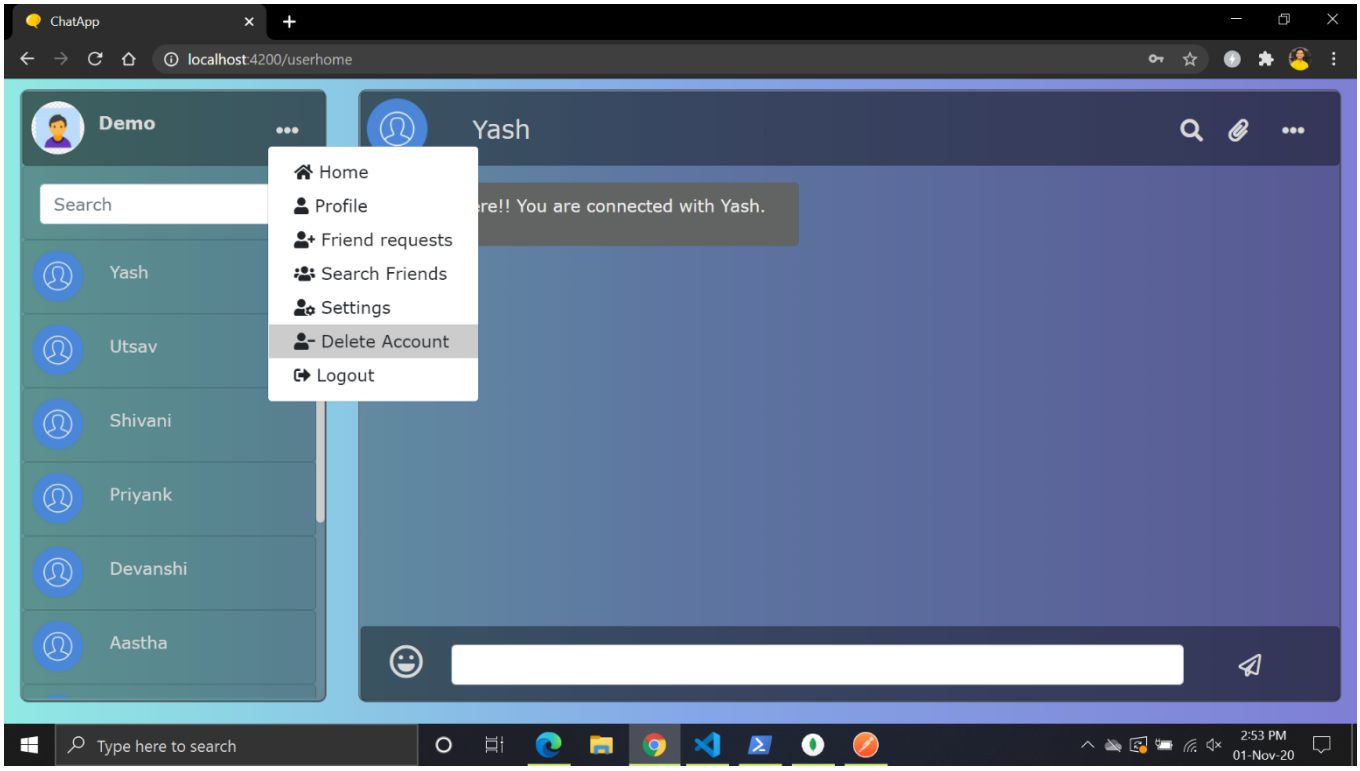

ChatApp
localhost:4200/userhome
Yash
Search
Utsav
Shivani
Priyank
Devanshi
Aastha
Shlok
Demo
Hii demo..
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
That's great !!!!...Amazing isn't it?
Type here to search
2:52 PM
01-Nov-20

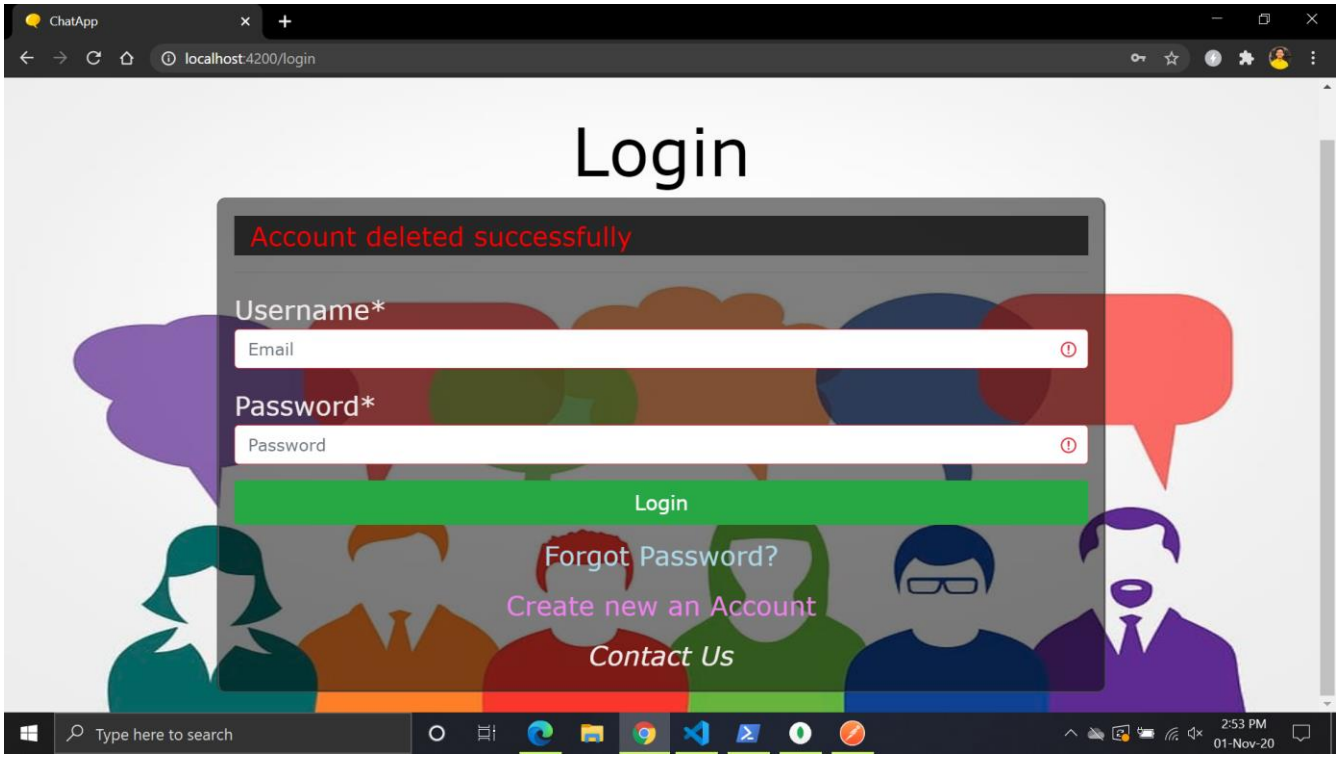

ChatApp
localhost:4200/userhome
Yash
Search
Utsav
Shivani
Priyank
Devanshi
Aastha
Shlok
Demo
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
That's great !!!!...Amazing isn't it?
Yes I know
Type here to search
2:52 PM
01-Nov-20

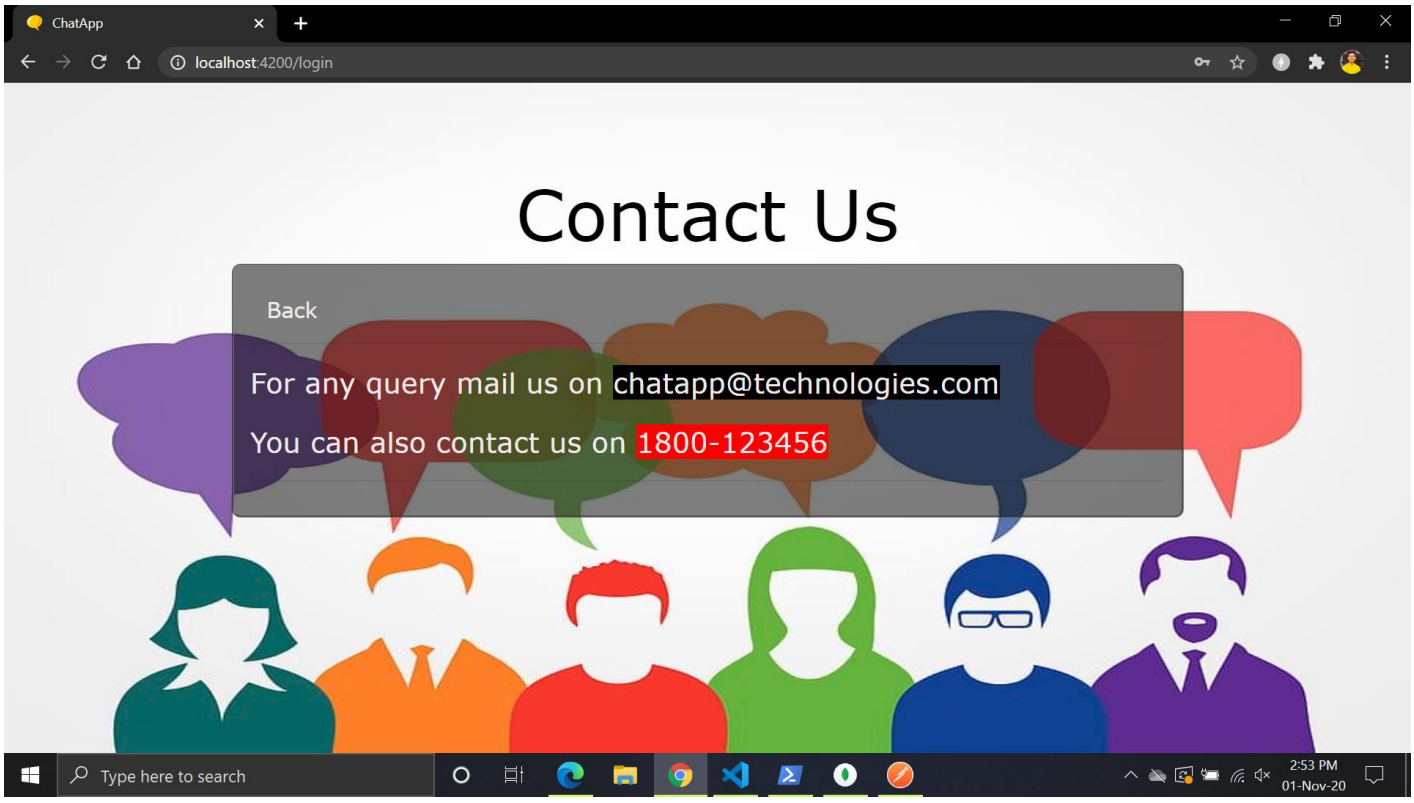

ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
You can send any language here
નોવેલ કોરોનાવાયરસ કે સાર્સ કોરોનાવાયરસ નામના વિષાણુને કારણે સમગ્ર વિશ્વમાં અને ભારતમાં ૨૦૧૯-૨૦માં આ રોગચાળો ફાટી નીકળ્યો છે. આ વાયરસ ચીનના વુહાન શહેરમાં ઉદ્ભવ્યો હતો અને ત્યારબાદ સમગ્ર વિશ્વમાં પ્રસર્યો છે.
That's great !!!!...Amazing isn't it?
Yes I know
Type here to search
2:52 PM
01-Nov-20

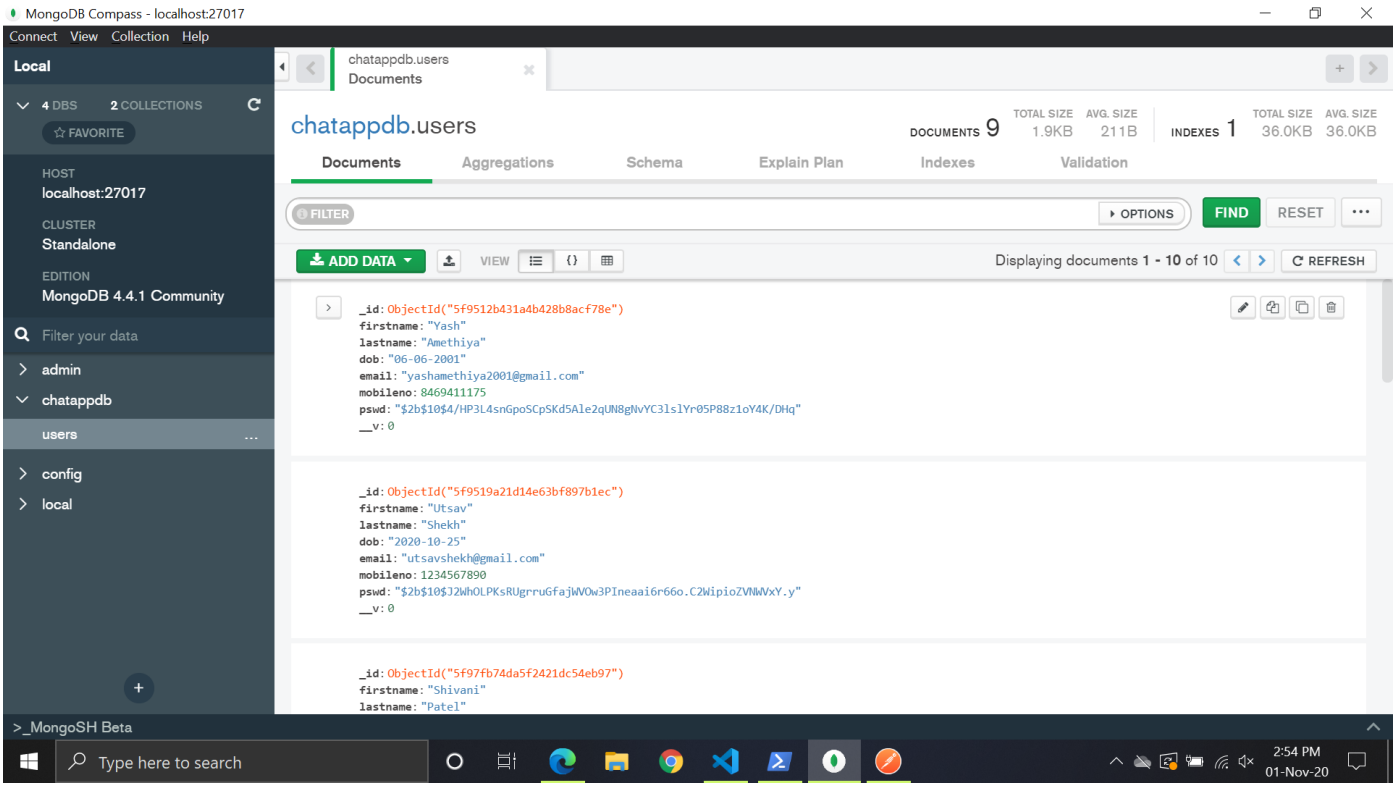

ChatApp
localhost:4200/userhome
Demo
Home
Profile
Friend requests
Search Friends
Settings
Delete Account
Logout
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Yash
That's great !!!!...Amazing isn't it?
Yes I know
Type here to search
2:53 PM
01-Nov-20


ChatApp
localhost:4200/login
Login
Successfully Logged out!!
Username*
Email
Password*
Password
Login
Forgot Password?
Create new an Account
Contact Us
Type here to search
2:53 PM
01-Nov-20


ChatApp
localhost:4200/userhome
Demo
Search
Yash
Utsav
Shivani
Priyank
Devanshi
Aastha
Home
Profile
Friend requests
Search Friends
Settings
Delete Account
Logout
Yash
re!! You are connected with Yash.
Type here to search
2:53 PM
01-Nov-20


ChatApp
localhost:4200/login
Login
Account deleted successfully
Username*
Email
Password*
Password
Login
Forgot Password?
Create new an Account
Contact Us
Type here to search
2:53 PM
01-Nov-20


ChatApp
localhost:4200/login
Contact Us
Back
For any query mail us on chatapp@technologies.com
You can also contact us on 1800-123456
Type here to search
2:53 PM
01-Nov-20


MongoDB Compass - localhost:27017
Connect View Collection Help
Local
4 DBS 2 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.1 Community
Filter your data
admin
chatappdb
users
config
local
chatappdb.users
Documents
chatappdb.users
DOCUMENTS 9 TOTAL SIZE 1.9KB AVG. SIZE 211B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 10 of 10 REFRESH
_id: ObjectId("5f9512b431a4b428b8acf78e")
firstname: "Yash"
lastname: "Amethiya"
dob: "06-06-2001"
email: "yashamethiya2001@gmail.com"
mobileno: 8469411175
pswd: "$2b$10$4/HP3L4snGpoSCpSKd5Ale2qUN8gNvYC3lslYr05P88z1oY4K/DHq"
__v: 0
_id: ObjectId("5f9519a21d14e63bf897b1ec")
firstname: "Utsav"
lastname: "Shekh"
dob: "2020-10-25"
email: "utsavshekh@gmail.com"
mobileno: 1234567890
pswd: "$2b$10$J2WhOLPKsRUgrruGfajWVOw3PIneaai6r66o.C2WipioZVNWVxY.y"
__v: 0
_id: ObjectId("5f97fb74da5f2421dc54eb97")
firstname: "Shivani"
lastname: "Patel"
>_MongoSH Beta
Type here to search
2:54 PM
01-Nov-20

Conclusion

Login functionality was successfully implemented by the use of auth guard service and token generation and verification. The animation, contact details was successfully implemented using CSS, Angular and bootstrap. List all the users for chatting was successfully implemented by fetching the details from the database with node js and displayed on the browser using CSS and Angular. The opening of chat window UI without refreshing/reloading the page was done using Angular. The welcome message and Message sending and receiving was successfully implemented using socket.io and socket.io-client. The drop down menu UI was implemented with CSS, Angular and Bootstrap and the icons were imported from fontawesome Add and Delete User Profile details, one-to-one encrypted messages was successfully implemented using post requests and mongoose RestAPI.

Limitation and Future Extension

Limitations:

- User can only see or read the message if he/she is currently in that chat.
- Search functionality for searching user is not working.
- User doesn't receive any notification for new message.

Future Extension:

- We would wish to extend searching message functionality and message saving into database.
- We would also like to add a chatbot for answering FAQ.
- Sending friend request and uploading images can also be added.
- For more security, we would like to add Auth () API for login and registration.
- Reporting or Blocking user against some or violation ransom

Bibliography

W3School:

- **HTML5:** https://www.w3schools.com/html/default.asp
- **CSS:** https://www.w3schools.com/css/default.adp
- **JavaScript:** https://www.w3schools.com/js/default.asp
- **AJAX:** https://www.w3schools.com/js/js_ajax_intro.asp
- **Jquery:** https://www.w3schools.com/jquery/default.asp
- **XML:** https://www.w3schools.com/xml/default.asp
- **XSLT:** https://www.w3schools.com/xml/xsl_intro.asp
- **XSD:** https://www.w3schools.com/xml/schema_intro.asp
- **JSON:** https://www.w3schools.com/js/js_json_intro.asp
- **Angular:** https://www.w3schools.com/angular/default.asp

Bootstrap4:

https://getbootstrap.com/docs/4.3/getting-started/introduction/

Icons:

https://fontawesome.com/icons?d=gallery

Express.js:

https://expressjs.com/en/api.html

Node.js:

https://nodejs.org/en/

Mongo DB:

https://www.mongodb.com/

Socket.io: https://socket.io/

For Making ER diagram and Flow chart:

https://www.lucidchart.com/pages/examples/data-flow-diagram-software

Other References:

https://stackoverflow.com/

https://www.youtube.com/

https://medium.com/