Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering
B. Tech. CE Semester – V

Subject: (CE – 515) Advanced Technologies

Project Title : ShareNChat- online chat aplication

By:
Name : Bhensdadia Rahil R.
Roll No. : CE012
Student ID : 18CEUON021

Guided by:
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 “Chat Application” is the bonafide report of work carried out by

Bhensdadia Rahil R. (18CEUON021)

of the Department of Computer Engineering, Semester V, academic year 2020-21, under our supervision and guidance.

_____

Guide

Prof. Prashant M. Jadav
Assistant Professor,
Department of Computer Engineering,
Dharmsinh Desai University, Nadiad

_____

HOD

Dr. C. K. Bhensdadia
Head of the Department,
Department of Computer Engineering,
Dharmsinh Desai University, Nadiad

Abstract

Chat Application market is dominated by foreign companies and the data of users is at the mercy of the volatile rules and security of these companies. With increase in the value of data and international tensions among countries it is the need of time that Bharat use its own indigenous applications. ShareNChat is a Browser based chat application as a small step towards this movement. With this application users can easily register and communicate with other user in form of chat without any significant delay in the communication.

The application uses login functionality for user verification and passwords are stored in form of hashed passwords this increases the security of users accounts.

Contents

1. Introduction

This project is a web-application developed using NodeJS, MongoDB, AngularJS, Express JS. The application is designed for single type of user. Users can register themselves using unique email address and login to the application to chat with other user . The chats are stored in database and retrieved when user logs-in. The application works primarily on socket technology to send and receive chats in real time but in case the socket connection gets interrupted by network traffic there is a fallback to send chat using API call to send the chat to server .The API and socket logic for the application is implemented using NodeJS. Database used is MongoDB which is a NoSQL database. Angular framework is used for single page implementation of frontend, HTML, CSS is used for frontend designing.

2. Technologies used

Frontend : AngularJS , HTML , CSS, Bootstrap
Backend : NodeJS, Express JS, socket.io
Database used : MongoDB

HTML : Hyper Text Mark-up Language is used to create the main structure of a webpage

CSS : Cascading Style Sheets is used to define styles of HTML.

Bootstrap : It is a free and open source front end library for designing web applications and making it responsive.

AngularJS : AngularJS is a type script based front-end web development framework.

NodeJS : NodeJS is an open-source and cross-platform java script run-time environment which executes JavaScript code server-side.

Express JS : It is a web application framework, which is used for NodeJS. Express JS is mainly used for handling API requests and responses.

MongoDB : A No-SQL database and uses java script Object Notation-like documents with schemas.

Socket.io : A java script implementation of socket technology for state based continuous communication between client and server

3. Software Requirement Specifications

In this system, users are of one category :

Chat User

3.1 Manage User

- Register
  - input : user details
  - output : redirected to login page
- Login
  - input : user Id (email address), password
  - output : Chat Home page
- Delete user
  - input : user selection from profile
  - output : The user will be deleted and its chat will be deleted from database .
- Logout
  - input : User selection
  - output: log-out user and redirection to login page

3.2 Manage chat

- Send chat
  - input : user can send chat using chat box and send button
  - output : chat content
- View chat history
  - input : user selection from list of chats
  - output : chat will be displayed
- Select user to chat
  - input : from list of users, user can select any one to chat
  - output : user can chat with that user using textbox

4. Design

4.1. XML:

User:

```
<users>
        <user _id="4654213348497 ">
            <FName>User1</ FName >
            <LName>dummy1</ LName >
            <email>12345@gmail.com</email>
            <phone>12345</phone>
            <imageUrl>DFGH.png</imageUrl>
            <password>21365421adfdc</password>
            <lastSeen>2020-10-31T06:27:34.522Z</lastSeen>
        </user>
</users>
```

Message:

```
<messages>

        <message _id="1354682184">
            <senderEmail>user1@gmail.com</senderEmail>
            <receiverEmail>user2@gmail.com</receiverEmail>
            <messageText>
                <![CDATA[Hey There!!!]]>
            </messageText>
            <messageTime>2020-10-31T06:27:34.522Z</messageTime>
        </message>

        <message _id="1354682186">
            <senderEmail>user2@gmail.com</senderEmail>
            <receiverEmail>user1@gmail.com</receiverEmail>
            <messageText>
                <![CDATA[Hello User1]]>
            </messageText>
            <messageTime>2020-10-31T06:27:34.522Z </messageTime>
        </message>
</messages>
```

Relation:

```
<relations>
        <relation _id=”12321354”>
                <userEmail>user1@gmail.com </userEmail>
                <friendEmail>user1@gmail.com </friendEmail>
        </relation>
        <relation _id=”12321355”>
                <userEmail>user1@gmail.com </userEmail>
                <friendEmail>user2@gmail.com </friendEmail>
        </relation>
        <relation _id=”12321356”>
                <userEmail>user1@gmail.com </userEmail>
                <friendEmail> user3@gmail.com </friendEmail>
        </relation>
</relations>
```

4.2 DTD

User:

```
<!DOCTYPE user
[
   <!ELEMENT users (user+)>
   <!ELEMENT user
(FName,LName,email,phone,imageUrl,password,lastSeen)>
   <!ELEMENT FName (#PCDATA)>
   <!ELEMENT LName (#PCDATA)>
   <!ELEMENT email (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
   <!ELEMENT imageUrl (#PCDATA)>
   <!ELEMENT password (#PCDATA)>
   <!ELEMENT lastSeen(#PCDATA)>
   <!ATTLIST _id ID#REQUIRED>
]
>
```

Message:

```
<!DOCTYPE messages
[
   <!ELEMENT messages (message+)>
```

```
   <!ELEMENT message (senderEmail,receiverEmail,messageText,messageTime)>
   <!ELEMENT senderEmail (#PCDATA)>
   <!ELEMENT receiverEmail (#PCDATA)>
   <!ELEMENT messageText (#PCDATA)>
   <!ELEMENT messageText (#PCDATA)>
   <!ATTLIST message _id ID #REQUIRED>
]
>
```

Relation:

```
<!DOCTYPE relations
[
   <!ELEMENT relations (relation+)>
   <!ELEMENT relation (userEmail,friendEmail)>
   <!ELEMENT userEmail (#PCDATA)>
   <!ELEMENT friendEmail (#PCDATA)>
   <!ATTLIST relation _id ID #REQUIRED>
]
>
```

4.3 XSD

User:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="users">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="user"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="user">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="FName"/>
        <xs:element ref="LName"/>
        <xs:element ref="email"/>
        <xs:element ref="phone"/>
```

```
      <xs:element ref="imageUrl"/>
      <xs:element ref="password"/>
      <xs:element ref="lastSeen"/>
    </xs:sequence>
    <xs:attribute name="_id" use="required" type="xs: NMTOKEN "/>
  </xs:complexType>
 </xs:element>
 <xs:element name="FName" type="xs:NCName"/>
 <xs:element name="LName" type="xs:NCName"/>
 <xs:element name="email" type="xs:string"/>
 <xs:element name="phone" type="xs:integer"/>
 <xs:element name="imageUrl" type="xs:NCName"/>
 <xs:element name="password" type="xs:NMTOKEN"/>
 <xs:element name="lastSeen" type="xs:NMTOKEN"/>
</xs:schema>
```

Message:

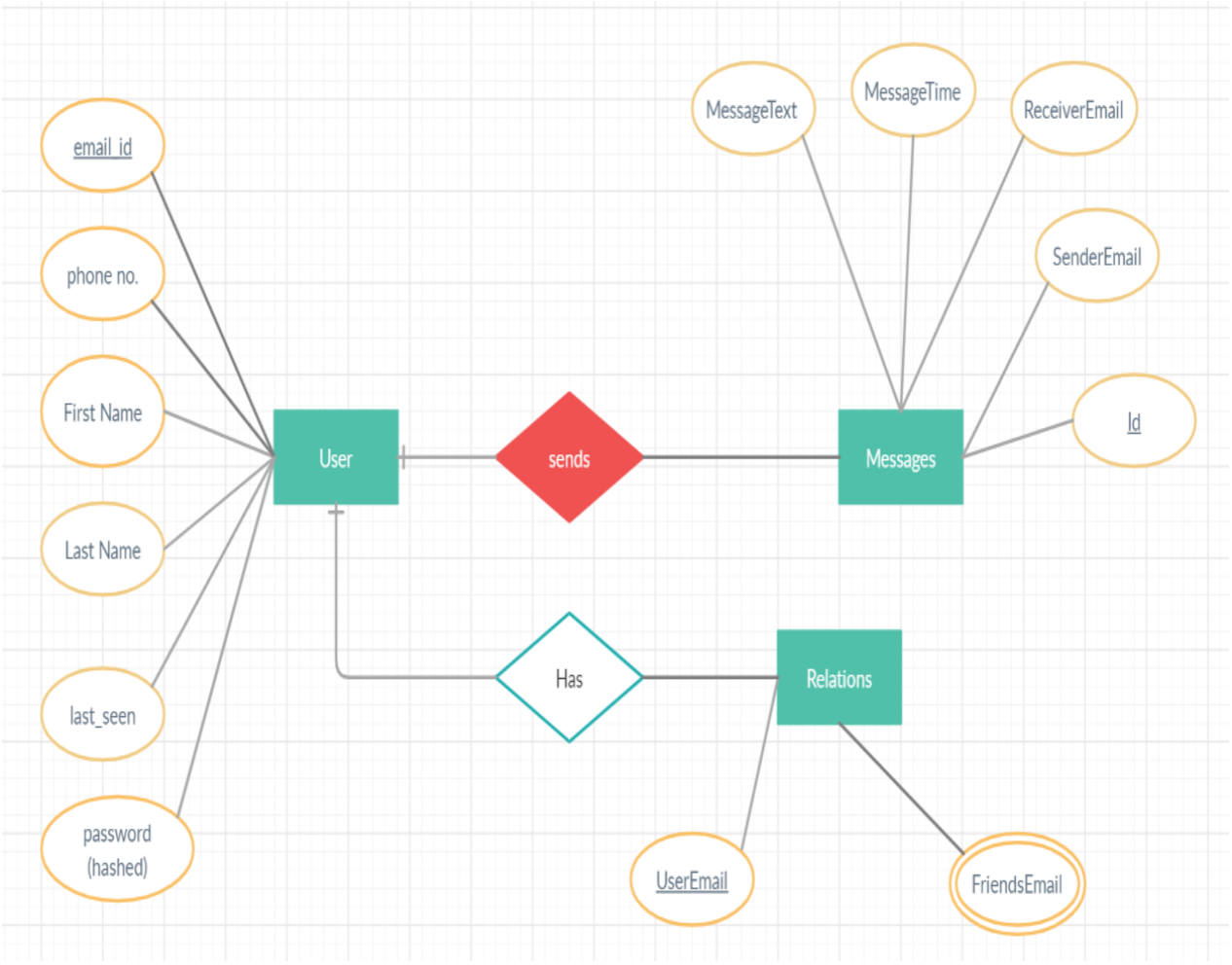
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
 <xs:element name="messages">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="message"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="message">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="senderEmail"/>
    <xs:element ref="receiverEmail"/>
    <xs:element ref="messageText"/>
    <xs:element ref="messageTime"/>
   </xs:sequence>
   <xs:attribute name="_id" use="required" type="xs:NMTOKEN"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="senderEmail" type="xs:string"/>
```

```
  <xs:element name="receiverEmail" type="xs:string"/>
  <xs:element name="messageText" type="xs:string"/>
  <xs:element name="messageTime" type="xs:NMTOKEN"/>
</xs:schema>
```

Relation:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="relations">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="relation"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="relation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="userEmail"/>
        <xs:element ref="friendEmail"/>
      </xs:sequence>
      <xs:attribute name="_id" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="userEmail" type="xs:string"/>
  <xs:element name="friendEmail" type="xs:string"/>
</xs:schema>
```

4.4 ER Diagram



5. Data Dictionary

User:

Sr.	Field Name	Data Type	Width	Required	Unique	PK/FK	Description
1	_id	Varchar	20	Yes	Yes	PK	AutoGenerate
2	FName	Varchar	20	Yes	No		
3	LName	Varchar	20	Yes	No		
4	Phone	Integer	15	Yes	No		
5	Email	Varchar	50	Yes	Yes		
6	Password	Varchar	50	Yes	No		
7	imageUrl	Varchar	200	No	No		
8	lastSeen	Varchar	15	Yes	No		

Message:

Sr.	Field Name	Data Type	Width	Required	Unique	PK/FK	Description
1	_id	Varchar	20	Yes	Yes	PK	AutoGenerate
2	SenderID	Varchar	50	Yes	No	FK	Refer User
3	ReceicerId	Varchar	50	Yes	No	FK	Refer User
4	MessageText	Varchar	500	Yes	No		
5	MessageTime	Varchar	20	Yes	No		

Relation:

Sr.	Field Name	Data Type	Width	Required	Unique	PK/FK	Description
1	_id	Varchar	20	Yes	Yes	PK	AutoGenerate
2	**UserId**	**Varchar**	**50**	**Yes**	**No**	**FK**	Refer User
3	**FriendId**	**Varchar**	**50**	**Yes**	**No**	**FK**	Refer User

6. Implementation Details

a. Modules created

- User Login/Register Module :
  - User login and verification
  - User registration and validation
  - Delete User on API call
- Message module :
  - Messages are transferred using socket and API to and from client to server
  - Messages are stored in database
  - Deletion of messages upon user deletion
- Socket Module:
  - Connect with client
  - Register client socket with client Email
  - Send acknowledgement
  - Receiver messages and relay those to recipients

b. Major Functionalities

API requests for :

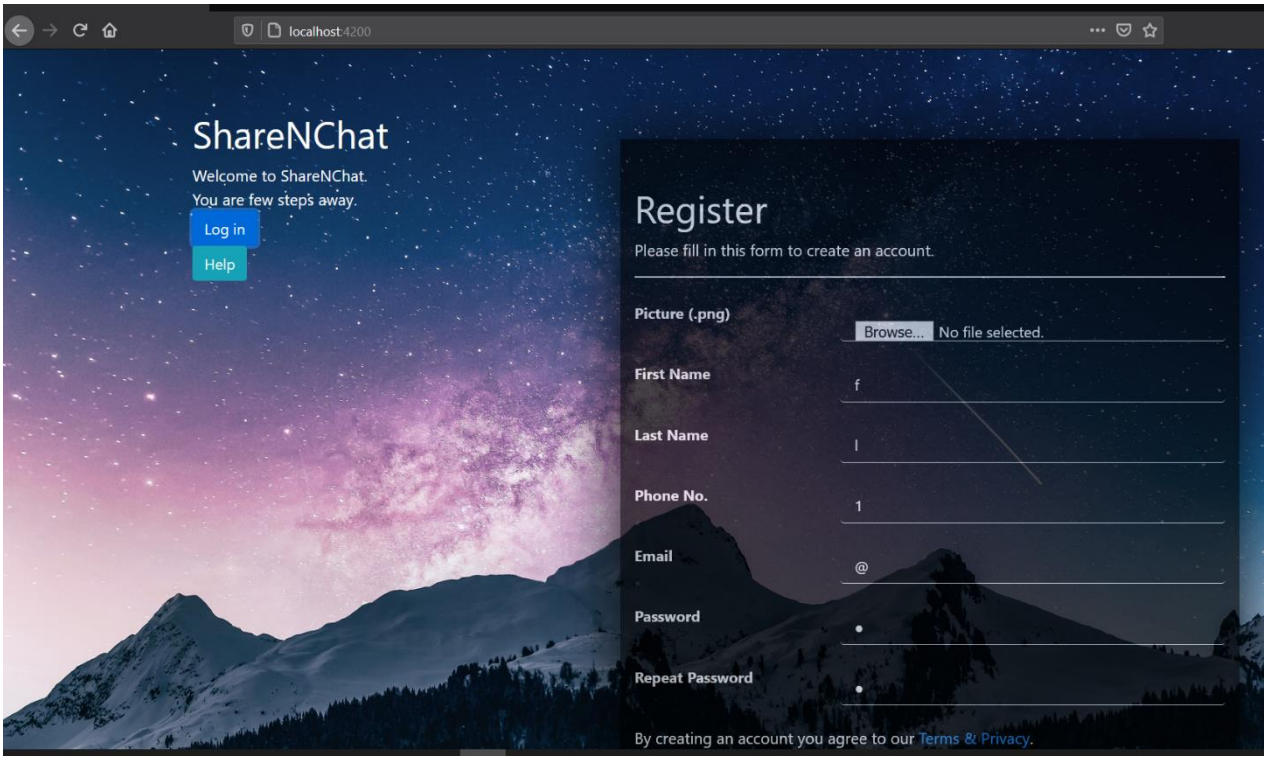
- User Register and validation of information and add them in database
- User Login and verification
- Getting user details by user Id
- Getting user list for user Id
- Deleting user by user Id
- Sending message using API call
- Deletion messages using message Id
- Getting message by user Id
- Sending socket connection request
- Sending user Id for naming the socket connection
- Sending message using socket
- Receiving messages from other users

7. Testing

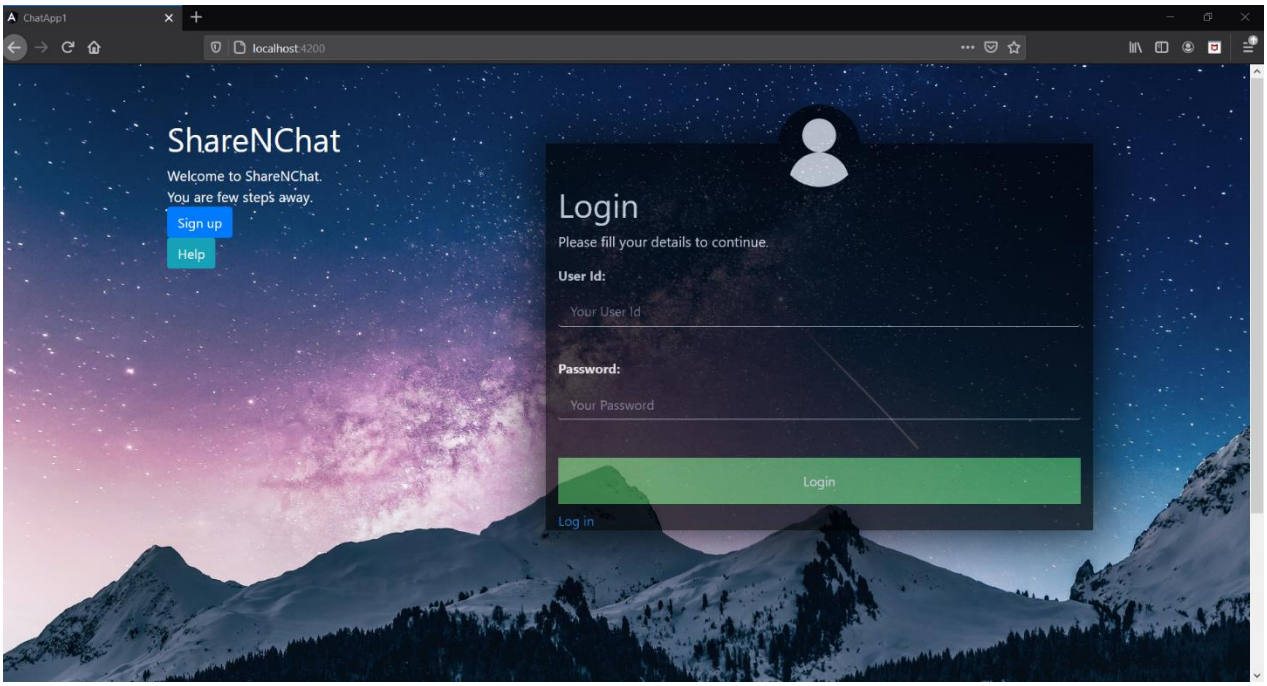
The application server was tested using “Postman” in order to fix bugs during development.

8. Screen shots

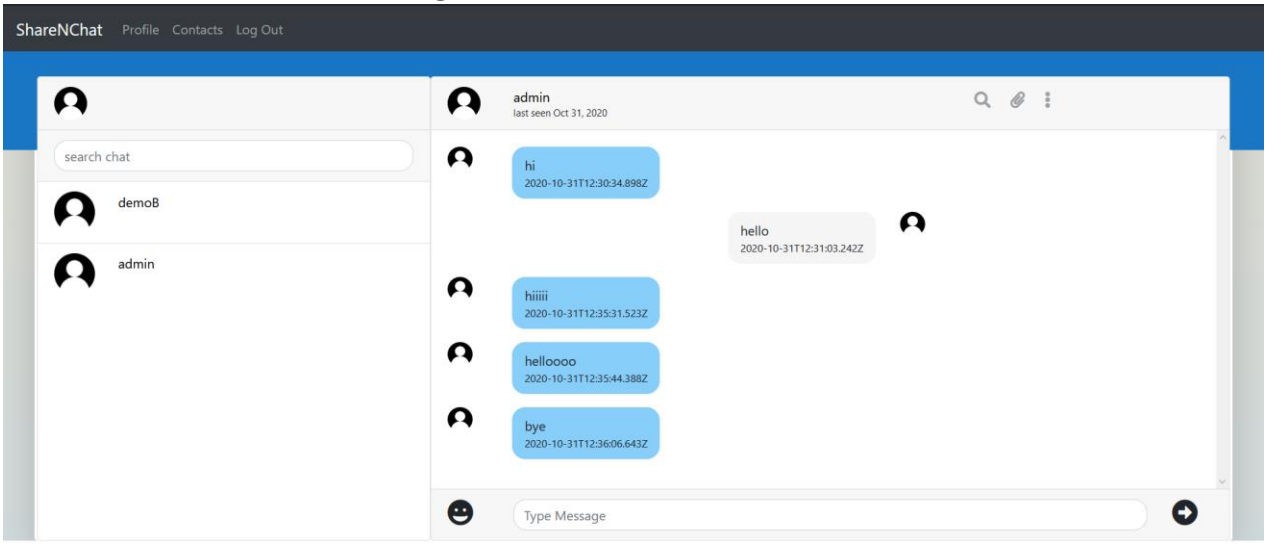
Registration screen



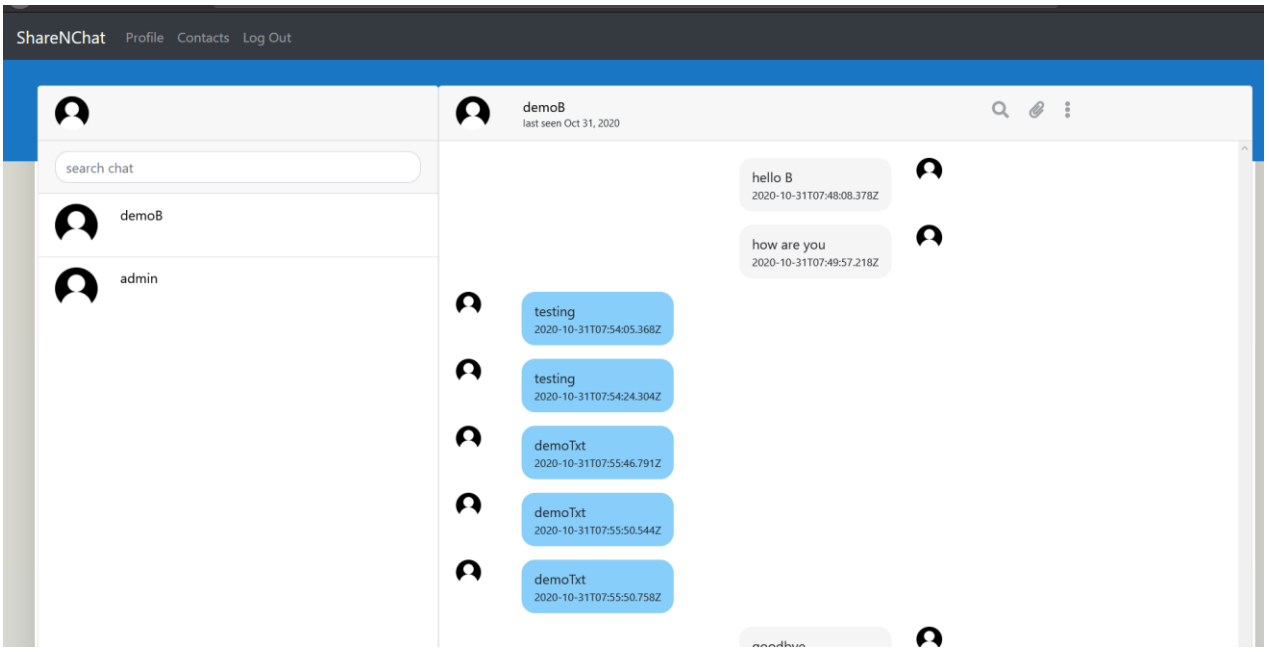
Login Screen



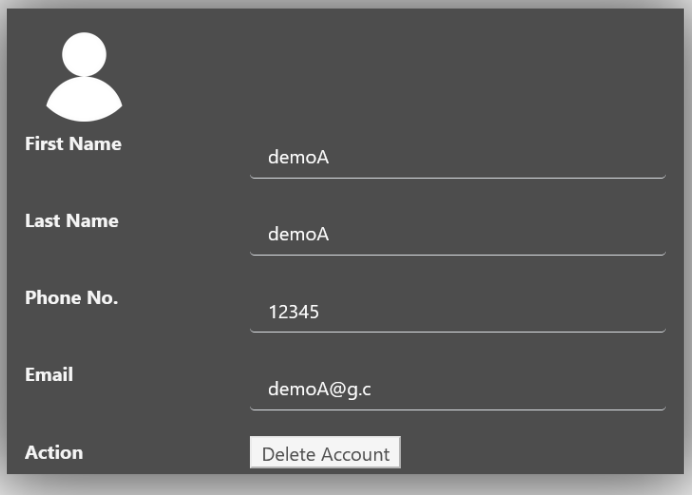
Home Scree after successful login



Home screen on clicking on chats with user named admin



Profile Screen



9. Conclusion

After understanding the project definition and modules thoroughly, following functionalities were successfully implemented :

- User registration with all validations
- User login and authentication
- User receiving chat history
- User sending messages
- User receiving the messages
- User view profile

- User Logout

10. Limitation and Future extensions

- Adding friends to contact list is remaining
- Adding functionality of message read acknowledgement
- Sharing images and documents
- Creating chat groups

11. 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

Bootstrap4:
https://getbootstrap.com/docs/4.3/getting-started/introduction/

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

Fonts:
https://fonts.google.com/

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

Node.js:

https://nodejs.org/en/

Mongo DB:

https://www.mongodb.com/

Auth0 API:

https://auth0.com/

Other References:

https://stackoverflow.com/
https://www.youtube.com/