College of Engineering Pune
(An Autonomous Institute of the Govt. of Maharashtra)

Software Engineering Mini Project II

Third Year Computer Engineering

CO-convo-EP

Under the guidance of

Mrs. Rohini Y Sarode

Prajjwal Datir - 111803131

Ganesh Gitte - 111803139

Aditi Medhane - 111803177

TABLE OF INDEX

1. Abstract

In recent years, chat applications have evolved and made a major impact on the way we interact with each other because of many distinctive features they provide making our life easy. But being chat applications for the general public they aren't optimized for a specific purpose especially for colleges. Also the recent changes to the privacy policy of major chat apps like Whatsapp, the concerns about the privacy of our data are at peak, because of that many of us are looking for alternative chat applications.

Due to the improvement in technology, mobile phones and laptops have become one of the integral parts of our daily life. There are millions of smartphone users in this world and people tend to use their smartphone for accessing many applications including different chat applications.

So, we thought of making an application which will be focused to make the life of college students and teachers easy, productive. We will try to overcome different limitations of available chat applications and provide a secure and optimized chat application for college students and teachers. Providing features like real-time messaging, including text messages, etc., creating and managing groups with group category folders and a tree like structure to effectively navigate and visualize the groups.

Aims

- Providing Colleges, Schools and other Educational Institutes with a secure chat application.

- Increasing the productivity by optimizing the application according to the College/School workflow.

- Making the application available to use on different devices (android, ios, desktop, etc)

- Providing a user friendly interface focused on productivity.

Objectives

- Users should be able to securely chat to other people personally or in a group.

- Users should be able to create and manage groups of people in an efficient manner

- Allow anyone to use the application on any smartphone/desktop/laptop regardless of the operating system.

- Allow anyone using the application to increase their productivity and provide the ability to manage their chatting workflow

Functional requirements

Being a web based application, users will be able to use the software on any device via web browsers to chat with others. Personal and group chatting are the main functions of the software. For our project we will be using Web Development and SocketIO.

- Frontend : Frontend is done by using the open source ReactJS library from facebook with material UI for some of the components and tailwind CSS to style the interface.
- Backend : For backend we used the ExpressJS framework which provides simple API's for building backends along with NodeJS which is event-driven JavaScript runtime.
- Database : MongoDB Atlas as the database for our chat application
- Testing : Postman and Jest Framework
- Development Environment : VSCode

Summary

- Design a Web Application using the MERN stack which will allow users to use the application in any web browser as a lightweight application to our device.
- Add features like filtering according to type of the group, Tree View of Official Groups.

2. Introduction

The CO-convo-EP application will allow Students And Teachers to be able to communicate in an effective manner to achieve maximum productivity.

This application will be focused to make the life of college students and teachers easy, productive and is available to be used on different devices whichever user prefers.

We will try to overcome different limitations of available chat applications and provide a secure and optimized chat application for college students and teachers.

3. Software Requirement Specification

1. Introduction

1.1 Purpose

The purpose of the CO-convo-EP application is to allow Students And Teachers to be able to communicate in an effective manner to achieve maximum productivity. It'll have features such as users will be able to chat with each other, user to user as well as group communication will be possible.

This application will be focused to make the life of college students and teachers easy, productive and is available to be used on different devices according to user preference.

We will try to overcome different limitations of available chat applications and provide a secure and optimized application for college students and teachers. Providing features like real-time messaging, including text messages, etc., creating and managing groups with group category folders and a tree like structure to effectively navigate and visualize the groups hierarchy.

We'll be discussing SRS in depth in the following points.

1.2 Document Conventions

UI	:	User Interface
UX	:	User Design
MERN	:	MongoDB, Express.js, ReactJS, NodeJS Frameworks
User	:	Person who is using the web-app on client side
Font	:	Arial
Full Stack	:	Tech stack which includes Front-end, Back-end & Database.

1.3 Intended Audience and Reading Suggestions

Developers Class	: Developer willing to build projects in Javascript Stack.
User Class	: Students in Universities
Teacher Class	: Teachers in Universities

1.4 Product Scope

Our web application will be made specifically for Universities & Students, to make their life more organized and productive.Our key features include Tree View & Filtering of Official Groups according to type of the group.

Our App will be Free and Open Source Software supporting Free software ideology. Teachers will have extra features to create an unofficial group by themselves and tag it to a year and branch. While students can only Join/View Official groups and can create(admin)/edit(admin)/view/join the Unofficial groups.

2. Overall Description

2.1 Product Perspective

CO-convo-EP is a secure chat application designed for Colleges, Schools and other Educational Institutes to increase simplicity and productivity of users. It could be useful when a user needs to exchange information in an efficient manner.

2.2 Product Functions

- **Personal and Group chatting** : Users will be able to chat with people in their contact list personally as well as in a group of people created by the user.
- **Tree View** : Users will get a tree-like interface to easily navigate to the desired group saving a lot of time and increasing productivity.
- **Group Categories** : The user can divide the groups into different categories for different teams or occasions or to create a workflow.

2.3 User Classes and Characteristics

- Typical users who will be benefited from this project will be from educational background. Ex. Faculties and Students
- Programmers who are exploring about Socket.io & Javascript framework
- Testers who are exploring Jest framework usage.

2.4 Operating Environment

The application is designed to work on any Android or IOS supporting Internet Explorer 9 and above(inclusive of all parallel versions of Chrome, Firefox, etc.)

2.5 Design and Implementation Constraints

CO-convo-EP is a progressive web application developed using MERN stack. Real-time communication required for communication of users will be done using socket.io.

Frontend is done by using the open source ReactJS library from facebook with Bootstrap for some of the components and CSS to style the interface.For backend ExpressJS framework will be used which provides simple API's for building backends along with NodeJS which is event-driven JavaScript runtime.

2.6 User Documentation

- User : Will be provided a Help section on the Home Page.
- Developer : An elaborative README will be provided which will describe system architecture and other docs. Along with that GitHub Wiki will be provided for reference.

2.7 Assumptions and Dependencies

- Users must have an active internet connection.
- Users must have a latest version of Web Browser on their computer/android/ios.

3. External Interface Requirements

3.1 User Interfaces 3

- Web Browser : As CO-convo-EP is a Web application, there is no need for dependencies except for a web browser.

3.2 Hardware Interfaces 3

- Android/ IOS : As the web application can be useful as a mobile web application as well.The general system that can run a web-browser is enough.
- Keyboard

3.3 Software Interfaces

- React : Front-end JS library to be used for frontend work
- MongoDB-Atlas : To be used to store databases
- Socket.io : To be used to integrate live communication and to provide a environment along with peers

Express : Back-end JS library to design web application and APIs

3.4 Communications Interfaces 3

- HTTP : A secure HTTP will be used to communicate between client and server which will ensure that the data is encrypted. Encryption will be done with the help of Signal encryption Algorithm.
- SMTP : This protocol will be used for user email verification while registering.

4. System features

[Priority 1 is lowest and 10 is highest]

4.1 Chatting

4.1.1 Description and Priority

Chatting is the main feature of our application. Our web application will allow users to use chatting features from any smartphone, tablet, laptop or desktop on any operating system MacOS, Windows, Linux, Android, ios, etc.

Sharing of text messages, different media will be done through this feature.

Priority : 10

4.1.2 Stimulus / Response sequences

1. First user logs in or creates an account.
2. Select the group / person they want to chat with.
3. Type message in the message box.
4. Click on send button
5. The message will be encrypted and sent from sender to server and then to the receiver.

4.1.3 Functional Requirements

Users will be able to chat with each other by using the chatting feature.

4.2 Tree view

4.2.1 Description and Priority

While using different chatting applications we often find them not organized as well as not optimized for college / school use.

This feature organizes the official college groups in a tree-like structure increasing productivity for students as well as teachers.

Priority : 9

4.2.2 Stimulus / Response sequences

1. Logged in users will see a Tree View button at the bottom of the home page.
2. After clicking on the Tree View button the user will be able to see a tree-like arrangement of buttons of group folders.
3. Users will be able to click on these buttons to navigate into containing group buttons and the groups themselves.
4. The end point of tree-like structure will take users to their desired group/chat.

4.2.3 Functional Requirements

The software will provide users with a tree-like interface to easily navigate to the desired group saving a lot of time and increasing productivity.

4.3 Group category folders

4.3.1 Description and Priority

Usually the groups, personal chats are all in the same list in many chat applications. The group category folders feature classifies different groups into an official group, unofficial group or any other user defined group folder. This helps to maintain a productive workflow.

Priority : 8

4.3.2 Stimulus / Response sequences

1. Logged in users will create group folders with name of choice.
2. Users will add all the related groups in these folders.
3. The group folder button will appear in the navigation tab.
4. On clicking the group folder all groups in that folder will appear as a list.
5. Users will be able to switch between different group folders.

4.3.3 Functional Requirements

The user can divide the groups into different categories for different teams or occasions or to create a productive workflow.

4.4 Security features

4.4.1 Description and Priority

Privacy is a major concern to many people using any application, especially social apps. So, there are security features to protect user privacy like the signal encryption.

Priority : 10

4.4.2 Stimulus / Response sequences

1. When users are using the application for chatting the sent text are end to end encrypted
2. They are decrypted at the user's side only to protect privacy.

4.4.3 Functional Requirements

The user need not worry about the privacy and data concerns and will be assured about any messages sent to official groups.

5. Other Non functional Requirements

5.1. Scalability :

Webapp should be able to provide instant messaging services to all users at any given time.

5.2. Privacy :

User information is protected from other users to protect privacy.

5.3. Performance:

Applications must be lightweight and must send messages instantly. A bug tracker will be available where users can report any bugs they have encountered so that can be fixed in the next release.

5.4. Security Requirements :

Users must create a username that stores their personal progress profiles. A unique username and password should be assigned to the user. The password should be hashed using the particular technique to ensure security.

6. Other Requirements

1. Internet - User needs to have internet connection to be able to send text messages and share media to other users and groups

4. Literature Review

4.1 Tabular representation

Year Of The Paper	Title Of The Paper	Journal/ Conference Details	Methodology used	Proposed idea	Advantages /Achieved Objectives In Paper	Disadvanta-ge /Limitations
2008	Design of a Secure Chat Application based on AES Cryptographic Algorithm and Key Management	Mathematical Methods, Computational Techniques,	Defined a proto Col for communi cation & designed A system for Handling user Passwords & keys.	Use symmetric Cryptographic algorithm (AES advan-ced system)	Overcome the Drawbacks of Classical Symmetric Cryptographic system	Users are Limited within The Local Area Network (LAN) Security faults According to Internet version Might be ntroduced.

2018	Developing an End-to-End Secure Chat Application	IJCSNS International Journal of Computer Science & Network Security	The proposed Chat application Is compared with other popular Chat applications based on require-ments as well as testing	List of require-ments and Proposed Architecture to achieve the goal of Building a end to end secure Application is made.	**XSalso20 Algorithm** ideal For mobile Devices Because of its High security, high performance & battery life.	Some of the Requirements In list may be irrelevant.
2017	Security analysis of end-to-end encryption in Telegram	2017 Symposium on Cryptography and Information Security, Naha, Japan	Analysis of security vulnerabilities in E2EE	Building Blocks of E2EE & MTProto	MTProto is faster	MTProto has Security flaws
2017	Security Analysis of Telegram	Hayk Saribekyan (hayks@mit.edu) , Akaki Margvelashvili (margvela@mit.edu)	Analysis of Telegram Architecture, Functionalities, And history.	Less data Should be Publicly Shown to All users.	Vulnerability can Give false positive result.	exploited a leak on user availability
2009	Impact of Text Messaging on Communication	Journal of Undergraduate Research at Minnesota State University	Two focus groups of college going students were asked different questions on text messaging vs face to face communication	To find if text messages displaced face-to-face communication	This study showed text messaging does displace face-to-face communication. However, text messaging is just a small part of displacing face-to-face communication	This study focused on students in the Arts & Humanities College therefore more research could be done with students in different colleges and on a different age group such as high school students or adults.
2016	Survey Analysis on	SOCIS, Indira Gandhi National	An open source software called Lime survey was	To analyze intensity of usage of	Most of the respondents use whatsapp for 15	The study was conducted in Delhi with 136

	the usage and Impact of Whatsapp Messenger	Open University, New Delhi	used to conduct online survey with form containing both close-ended and open-ended questions to access the demographics of users	Whatsapp and its popular services along with impact of using Whatsapp and to seek frequency and interactivity of Whatsapp among its users.	to 60 minutes daily, 64% people think whatsapp doesn't harm them	respondents which could be extended to more cities and more respondents. There is limited literature review available in Indian context.
2006	A study of Internet instant messaging and chat protocols	IBM T.J. Watson Research Center	The most recent Internet Messaging clients and protocols were analyzed	An overview of Internet Messaging protocols as exemplified by the three popular systems: AIM, MSN, and YMSG presented	Internet Messaging and Internet chat are here to stay, and will continue to evolve over time	Little is known about the technical aspects of commercial Internet Messaging and chat protocols, due to the closed proprietary nature of these systems
2020	REALIZATION OF NATIVE APPS USING PROGRESSIV E WEB APPS	Journal of Emerging Technologies and Innovative Research	The set of technologies and concepts related to progressive web apps were analyzed	Examine and evaluate the status quo of current possibilities comparing against cross-platform app development approaches on both technical and overarching aspects	Rendering improved user interface and incorporating the necessary PWA features is not quite difficult, Several native applications are likely to be replaced in the future,Several native applications are likely to be replaced in the future,	The Progressive Web App must be further tracked by more study and market acceptance

2019	Do Not Harm in Private Chat Apps: Ethical Issues for Research on and with WhatsApp	Westminster Papers in Communication and Culture	Ethics in research on Ethnographic Chat Groups	Guarantee full anonymisation to research subjects	Ethnographic groups secured. Ethics are followed.	Less accurate research is done.
2018	Progressive Web Apps: the Definite Approach to Cross-Platform Development?	51st Hawaii International Conference on System Sciences	Holistic introduction to PWAs	Progressive Web Apps (PWAs) can be the definite approach to cross-platform development?	PWA can contribute to a richer development experience, and eventually better apps	Some Features still under development.
2017	Progressive Web Apps: The Possible Web-native Unifier for Mobile Development	13th International Conference on Web Information Systems and Technologies (WEBIST 2017)	Comparing with Ionic Framework Hybrid App, React Native App	Potential for PWAs to become a unifier for web-native development without the use of cross-platform frameworks	PWA are more User Friendly, Faster, Secure. PWA can look, feel and act similar to native, hybrid and interpreted apps	Hardware and platform API limitations to PWAs
2018	Identifying Usability Issues in Instant Messaging Apps on iOS and Android Platforms	Hindawi Mobile Information Systems Volume 2018, Article ID 2056290	Evaluation to detect the main usability issues of instant messaging apps	Features are recommended which are for the benefit of the User	Improves Usability, User Friendly, May lead to higher economic benefits.	Requires time & effort implementing all the recommended features.

4.2 Conclusion that summarizes the literature review

The paper "*Developing an End-to-End Secure Chat Application*" is one that heavily favours the assumption of our project to propose a chat application that provides End-to-End security that lets users safely exchange private information with each other without worrying about data In addition to the protection of storage.

The paper "*Impact of Text Messaging on Communication*" discusses the point whether text messages displaced face-to-face communication or not by conducting surveys on students attending colleges.

The paper "*A study of Internet instant messaging and chat protocols*" gives an overview of Internet Messaging protocols.

The paper "*Security Analysis of Telegram*" is one that heavily focuses on how Less data should be publicly visible to all users.

The paper *"Identifying Usability Issues in Instant Messaging Apps on iOS and Android Platforms"* discusses the points and features which will be recommended for the benefit of the user in a messaging application.

4.3 Analysis

Works Against :

The 2009 paper "*Impact of Text Messaging on Communication*" discussed the issue with using different text messaging apps on face-to-face communication which is an essential part in everyday life. This study showed text messaging does displace face-to-face communication. So, we should not use the text messaging apps for a lot of time. However, text messaging is just a small part of displacing face-to-face communication as we can see in the recent past we want more face-to-face communication than we need text messaging.

The 2016 paper *"Survey Analysis on the Usage and Impact of Whatsapp Messenger*" is a research on usage of whatsapp and its different features by people of different age groups. The study shows that most of the users use whatsapp for 15 to 60 minutes daily and they believe that whatsapp doesn't harm them. The study also shows that the usage is more in adults and is increasing day by day.

Works For :

The paper *"Design of a Secure Chat Application based on AES Cryptographic Algorithm and Key Management"* is the one that mainly focuses on design and development a software application that provides secure real time communication based on symmetric cryptographic algorithms(AES Advanced Encryption System) and a management system for handling, distributing, safely storing & retrieving user passwords.
The aim of the application was to provide the infrastructure that allows groups of authenticated users to read messages that they exchange in pairs.The purpose of this whole subsystem was the safe storage of secret keys and passwords for each user and the access control function for the application.

The paper "*Developing an End-to-End Secure Chat Application*" is one that heavily favours the assumption of our project to propose a chat application that provides End-to-End security that lets users safely exchange private information with each other without worrying about

data In addition to the protection of storage. A list of requirements to make secure chat applications are presented in this paper. The proposed chat application is compared with other popular applications based on those requirements(Whatsapp, Viber, Telegram, Facebook Messenger) as well as testing is done as a proof for providing End-to-End security.

Neutral Works :

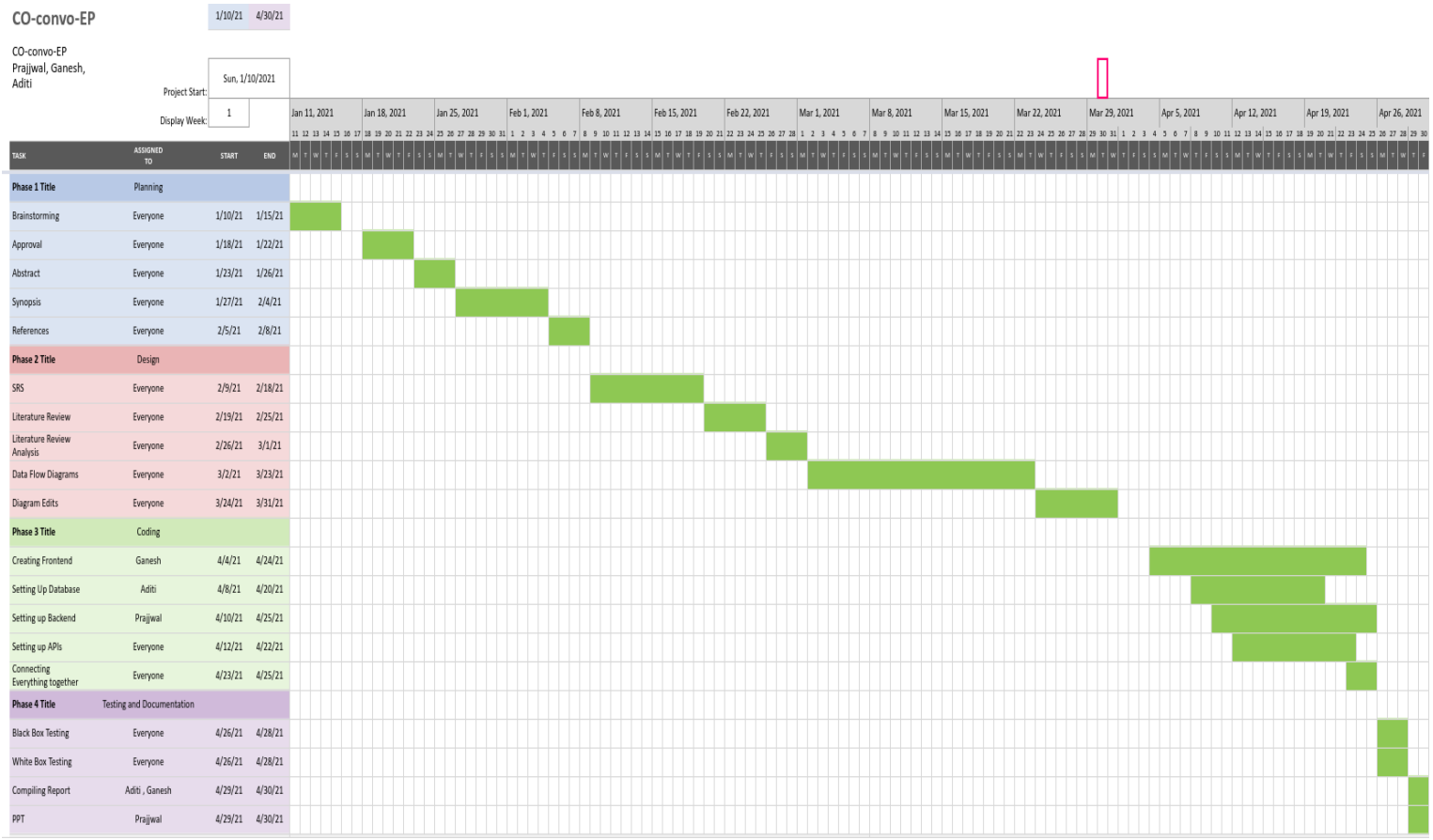
The 2017 paper ”*Progressive Web Apps: The Possible Web-native Unifier for Mobile Development”* is one that recognises potential for PWAs to become a unifier for web-native development without the use of cross-platform frameworks.It also focuses on main features such as user friendly, faster, secure.

The 2018 paper *“Progressive Web Apps: the Definite Approach to Cross-Platform Development?”* mainly focuses on how PWA can contribute to a richer development experience.

5.Proposed Statement

To create a web application that can be of help to the University/College community. Will feature a group filtering & tree view component as its main part along with other services like one-one chatting, authentication etc to further foster a sense of community.

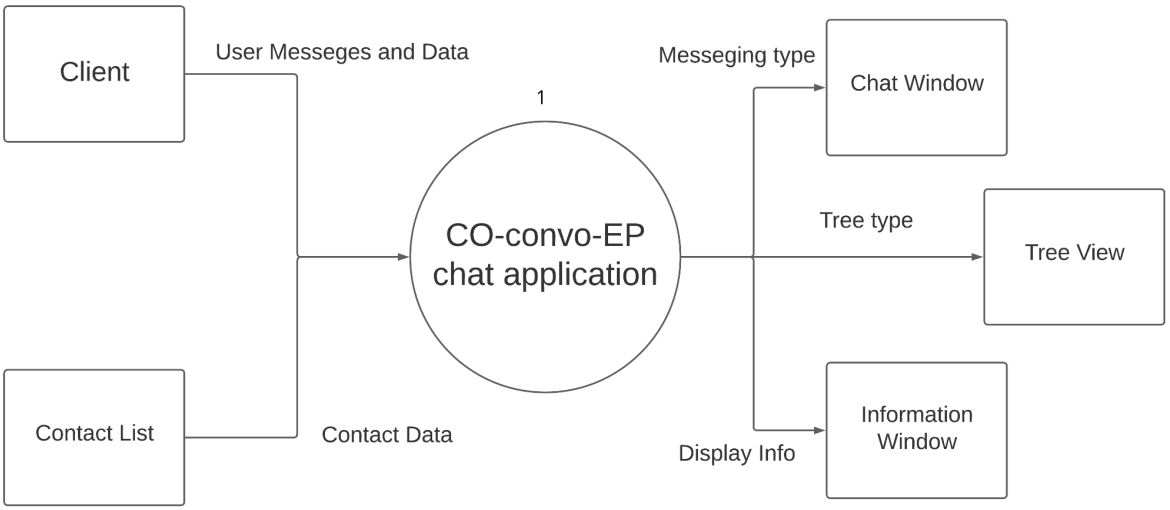
6.Planning



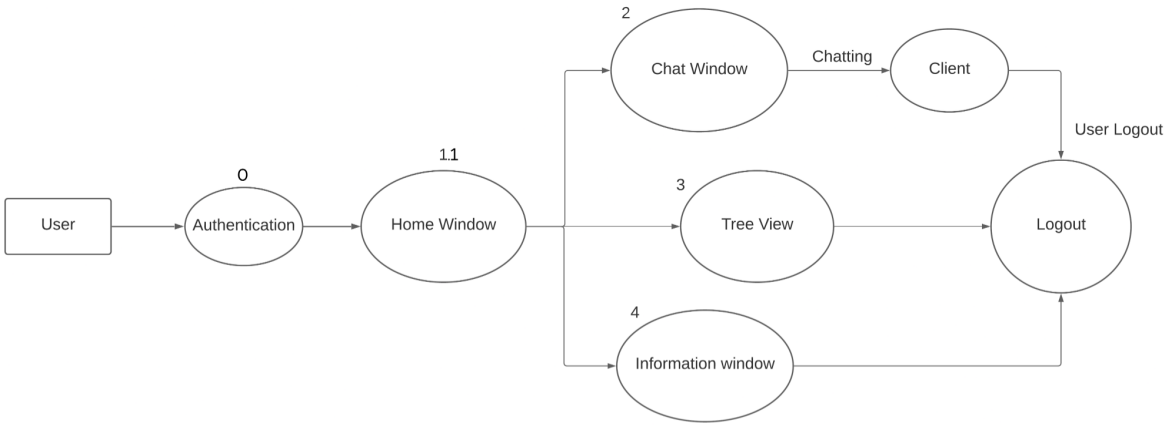
7.Design

7.1 Data Flow Diagram

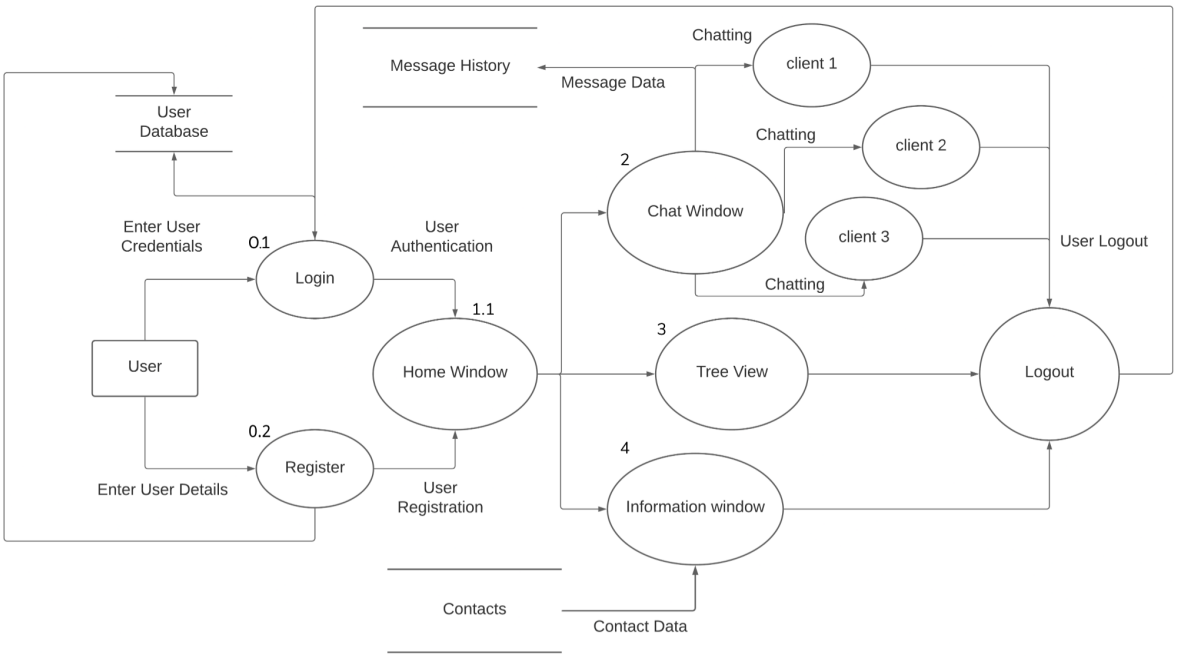
Level 0



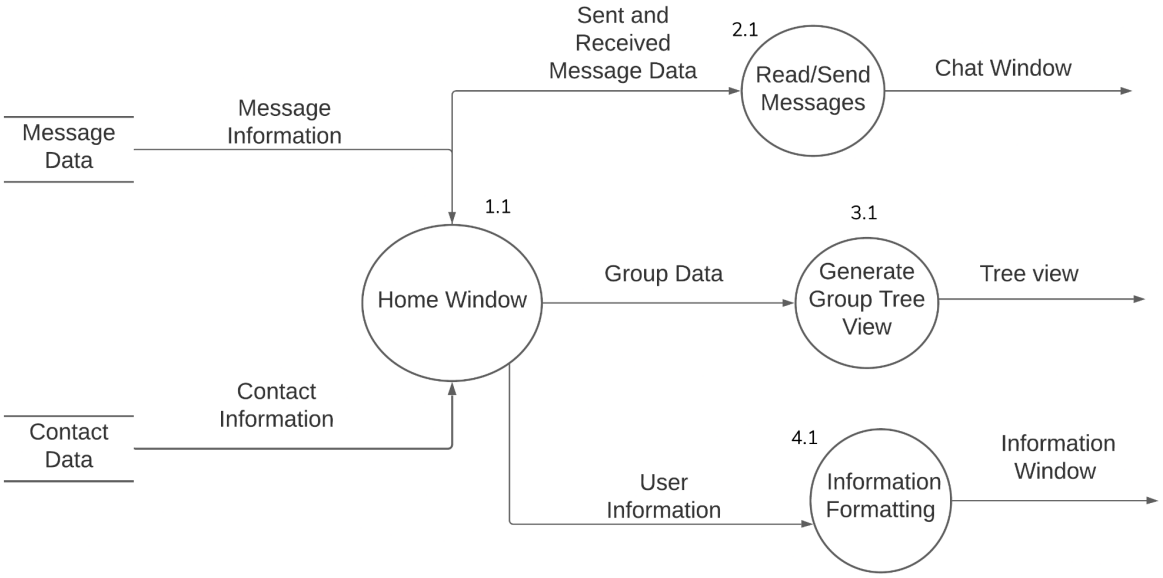
Level 1



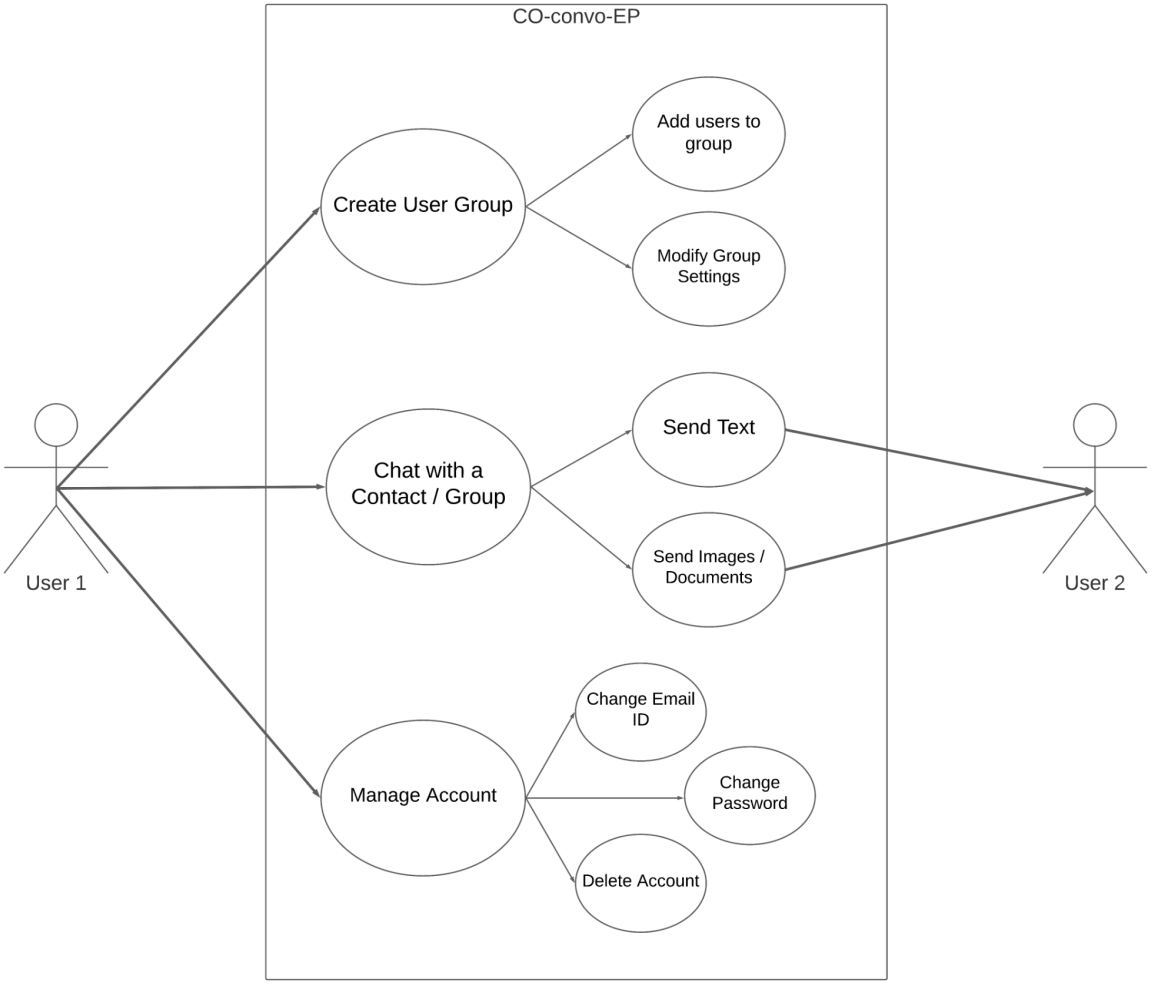
Level 2


Chatting
client 1
Message History
Message Data
User Database
Chatting
client 2
2
Chat Window
client 3
User Logout
Enter User Credentials
0.1
Login
User Authentication
Chatting
1.1
User
Home Window
3
Tree View
Logout
0.2
4
Register
Enter User Details
User Registration
Information window
Contacts
Contact Data

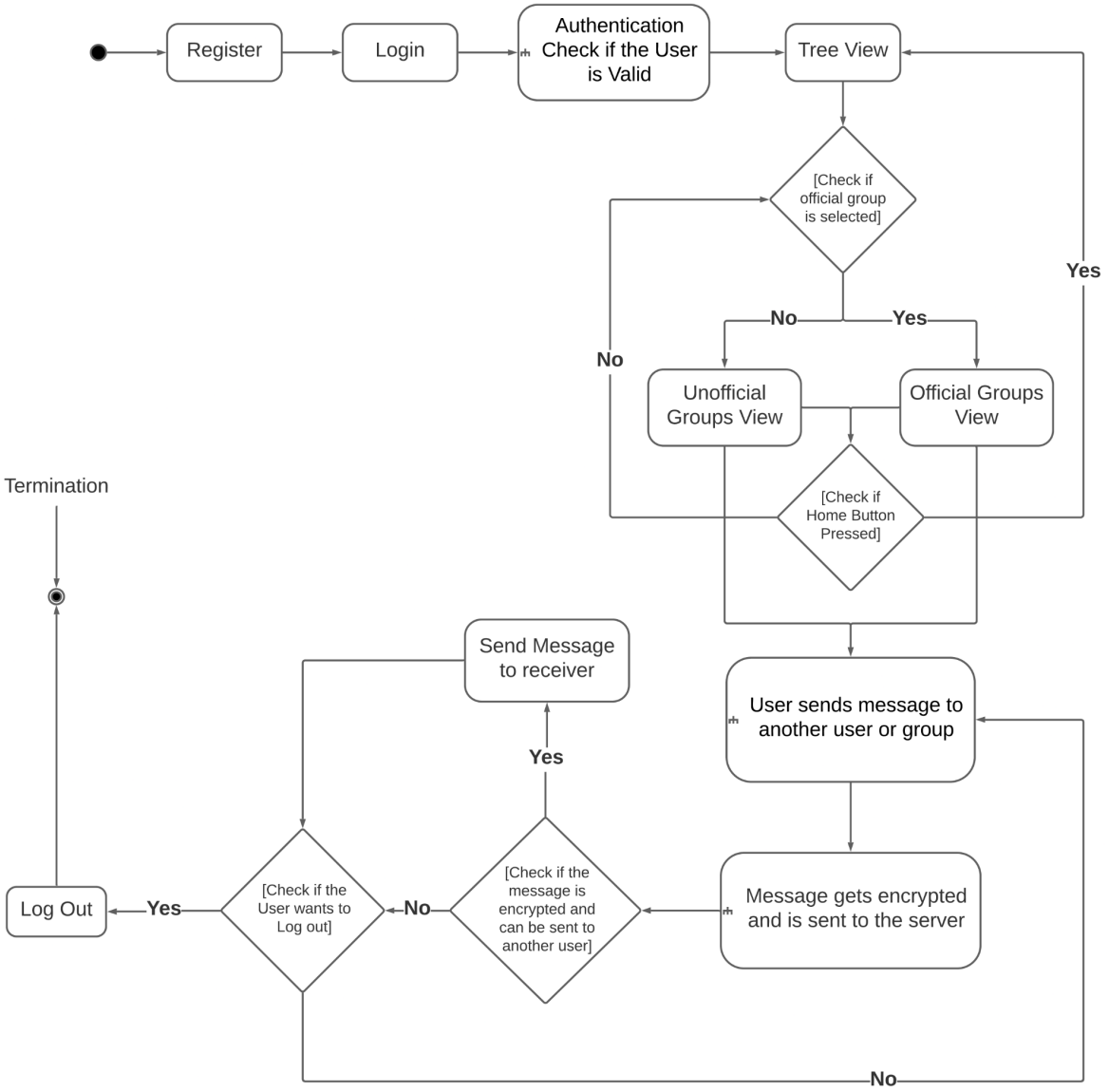
Level 3


Sent and Received Message Data
2.1
Read/Send Messages
Chat Window
Message Data
Message Information
1.1
3.1
Home Window
Group Data
Generate Group Tree View
Tree view
Contact Information
Contact Data
4.1
Information Window
User Information
Information Formatting

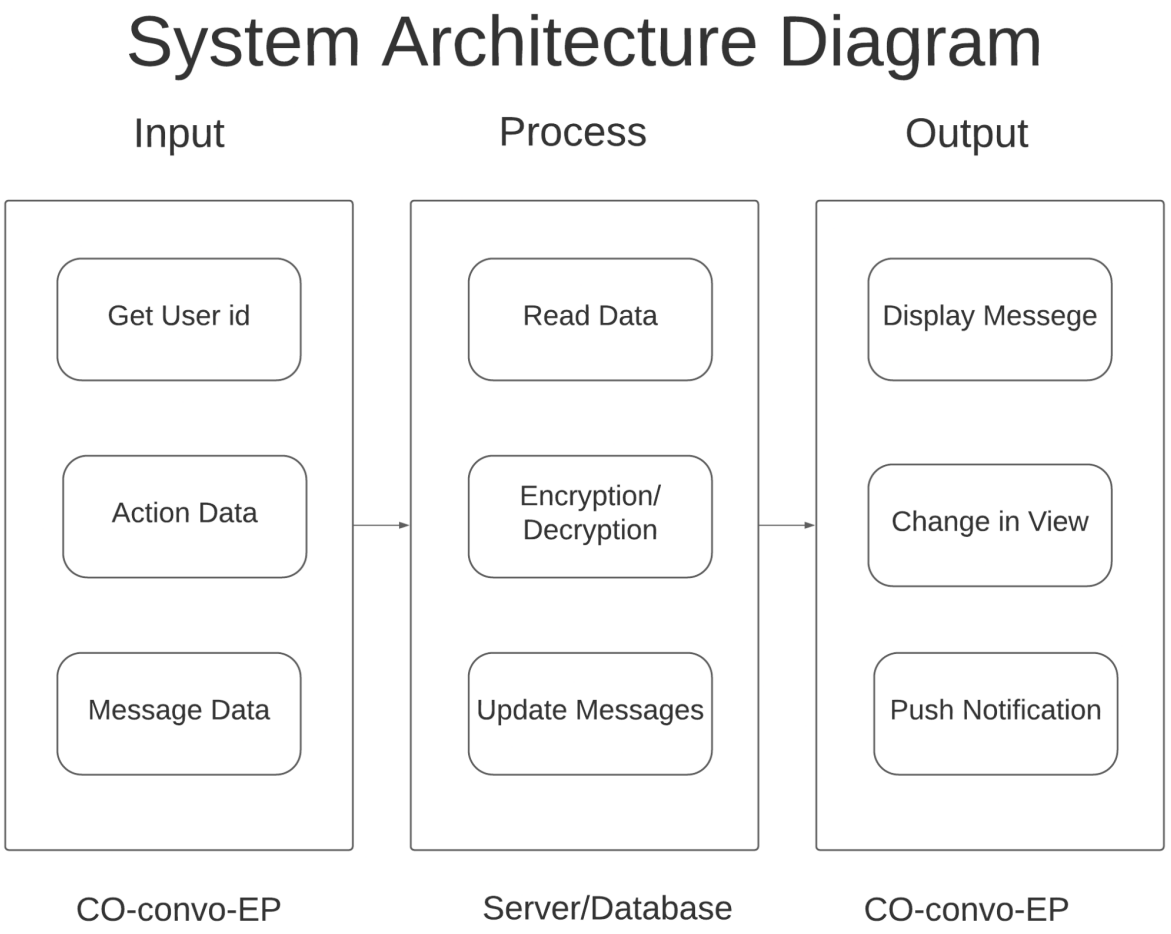
7.2 Use Case Diagram



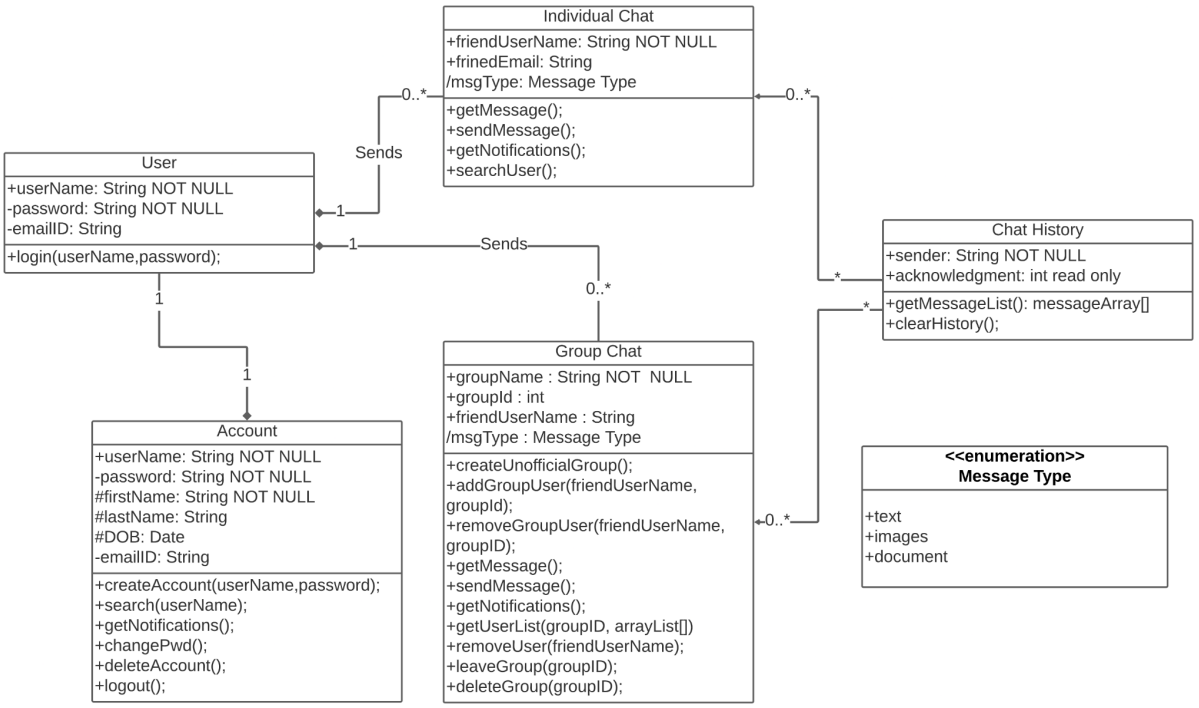
7.3 Activity Diagram



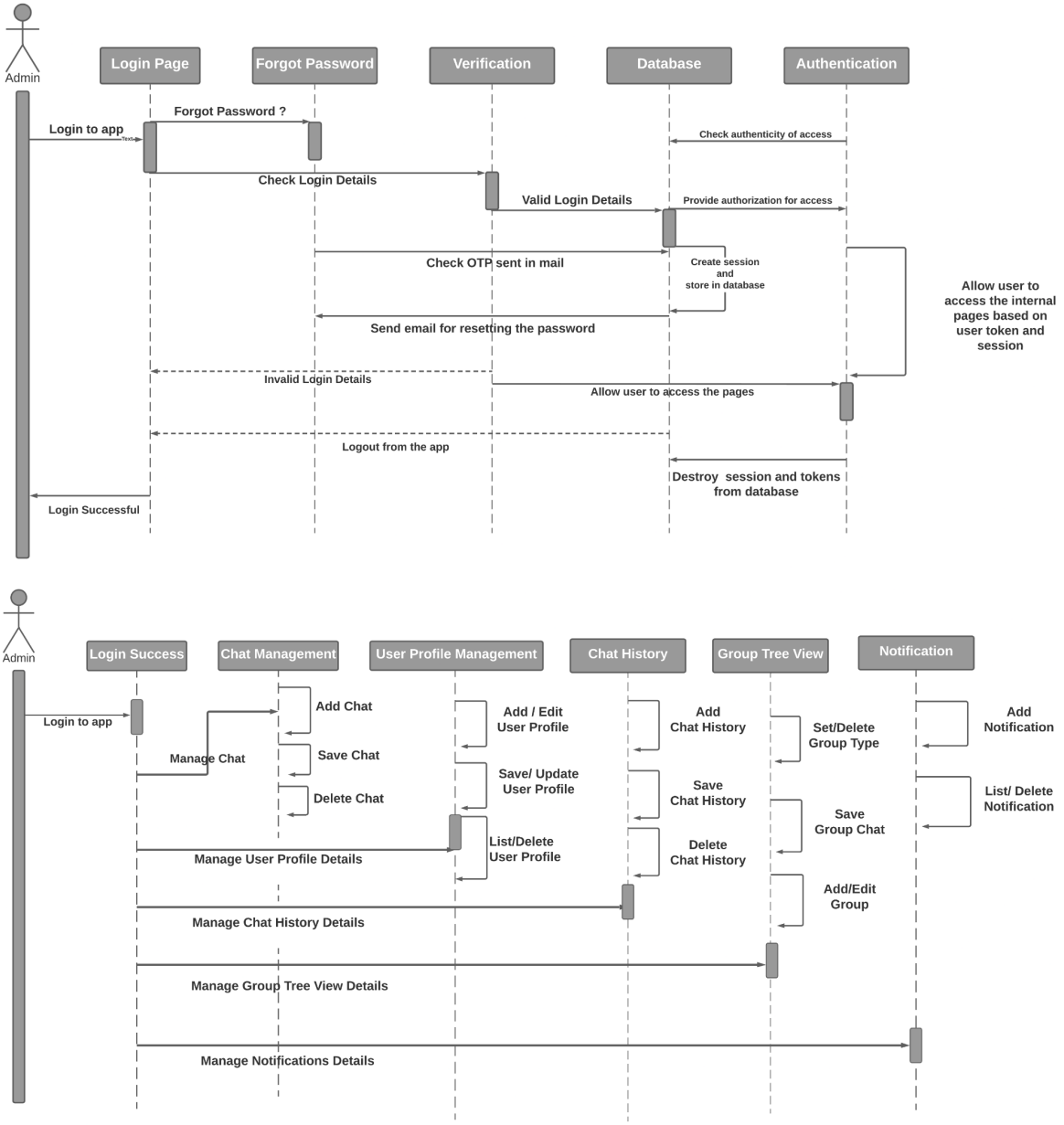
7.4 System Architecture



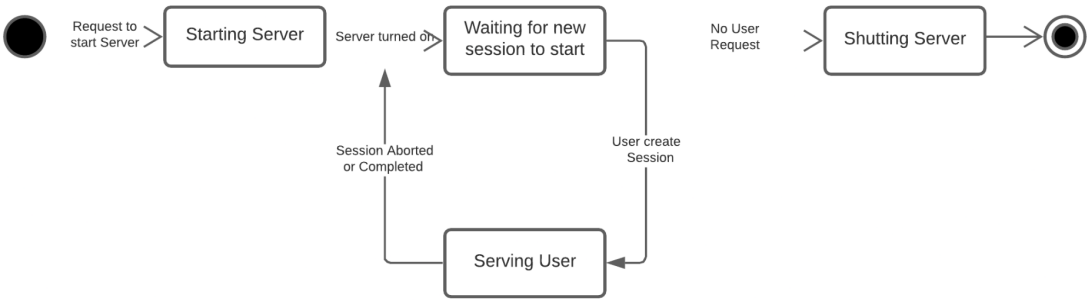
7.5 Class Diagram



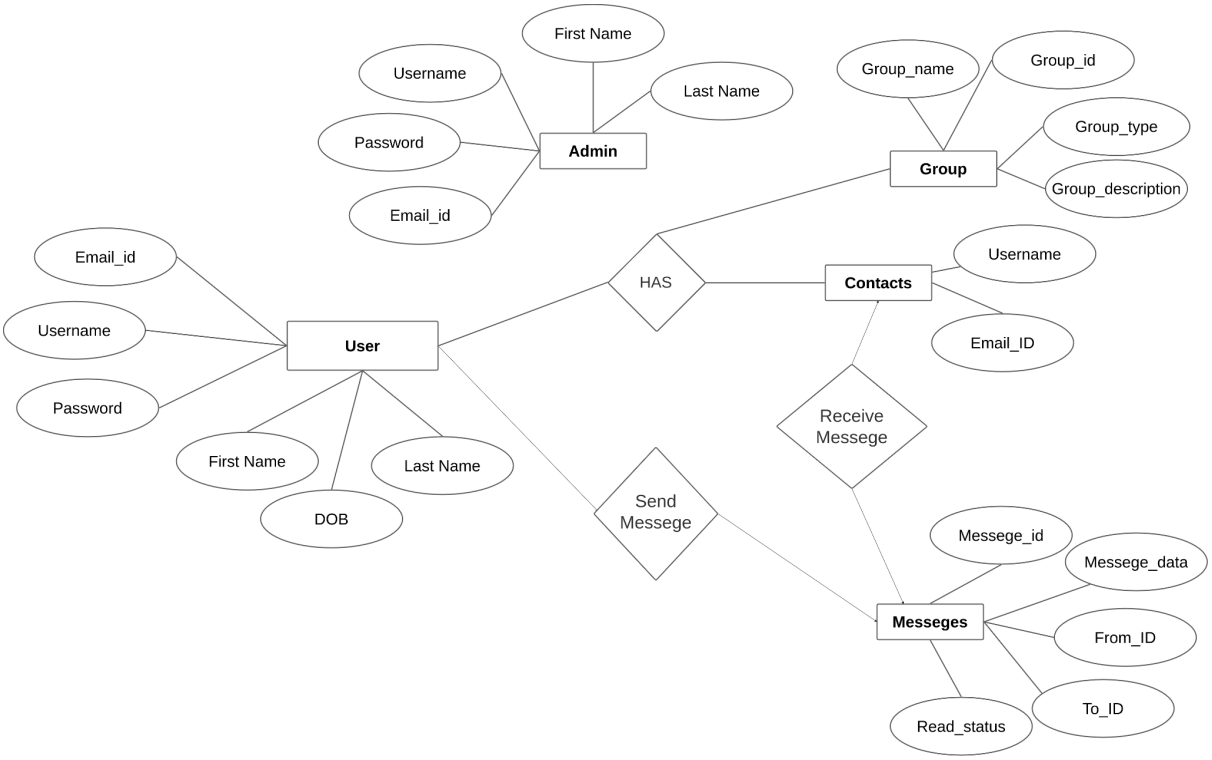
7.6 Sequence Diagram



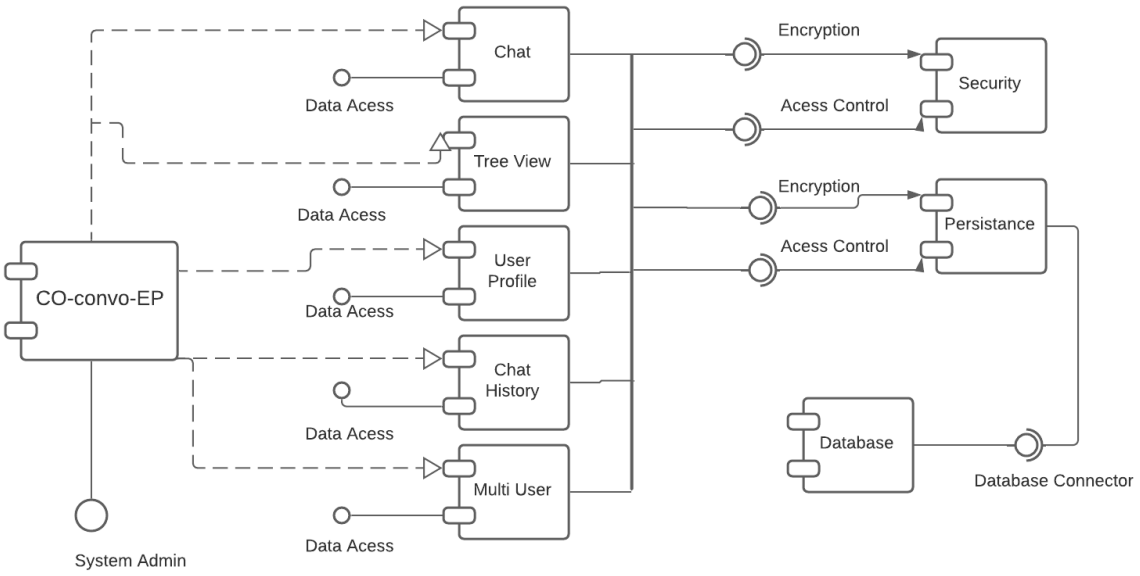
7.7 State Diagram



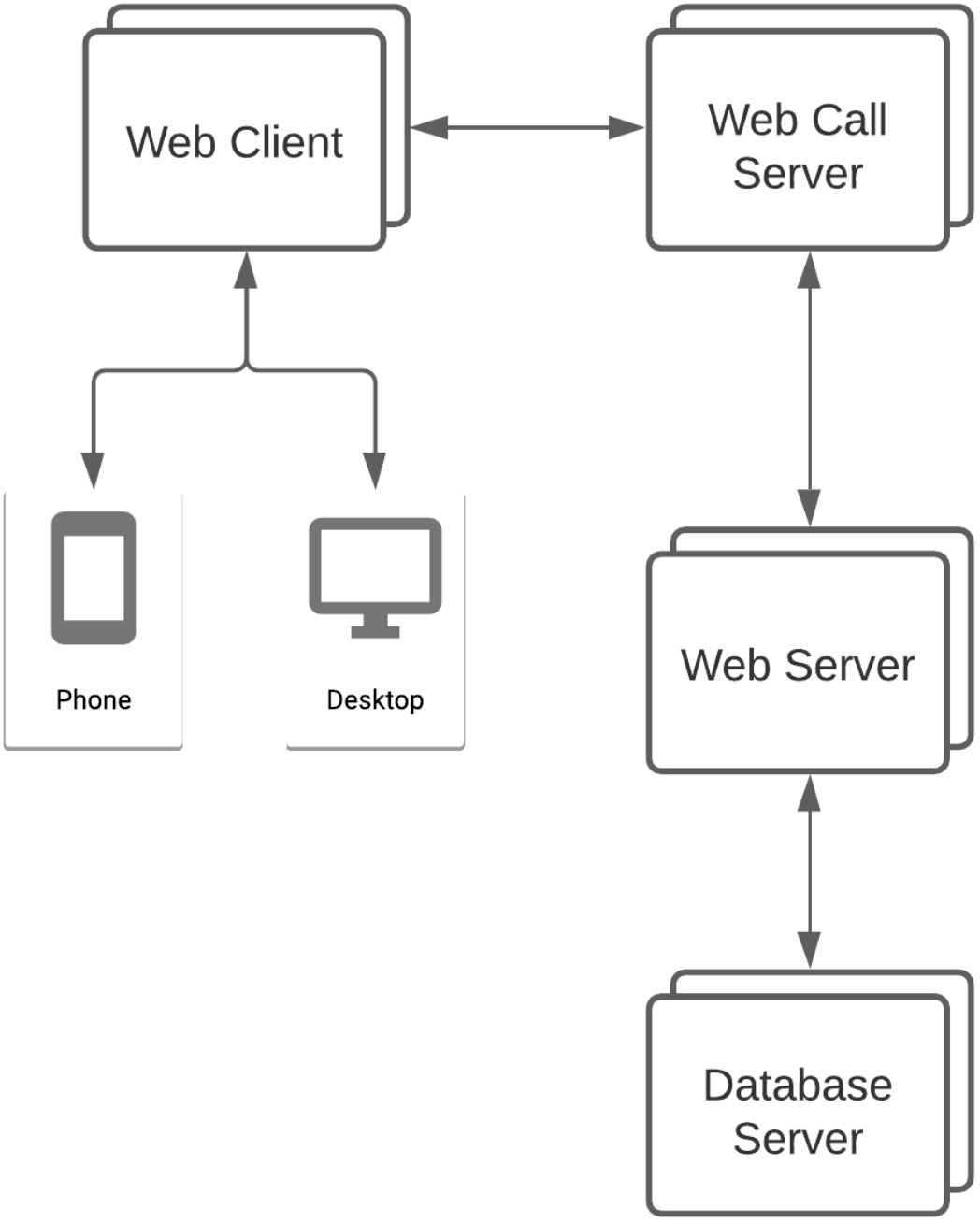
7.8 ER Diagram



7.9 Component Diagram

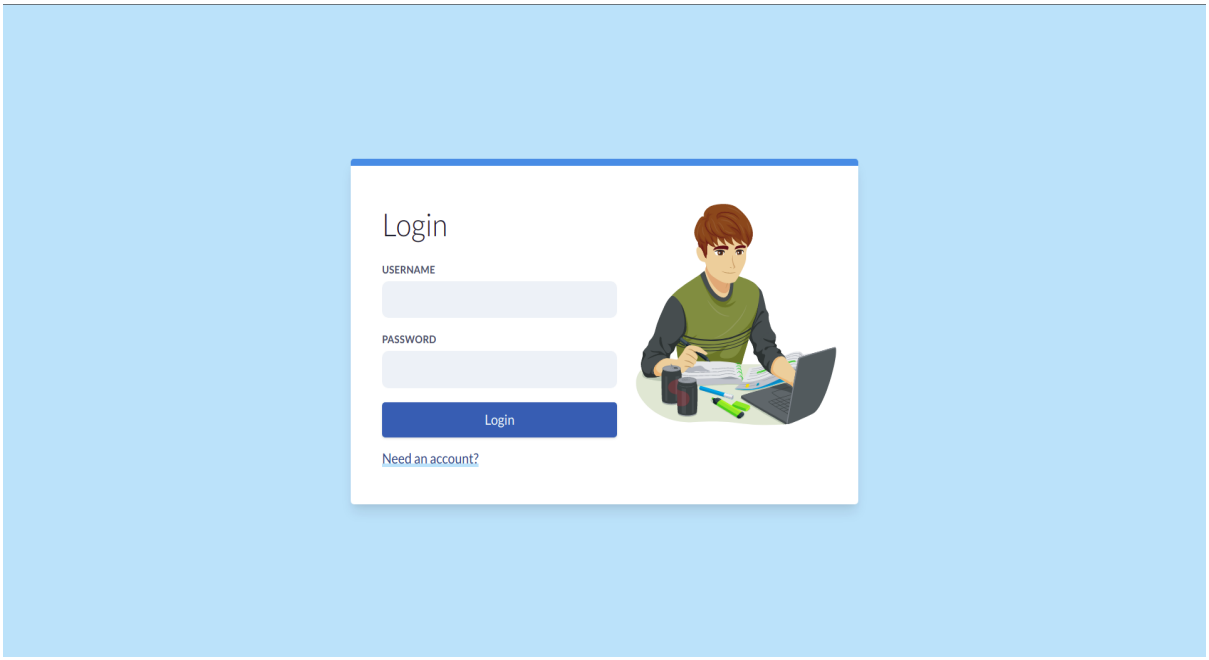


7.10 Deployment Diagram



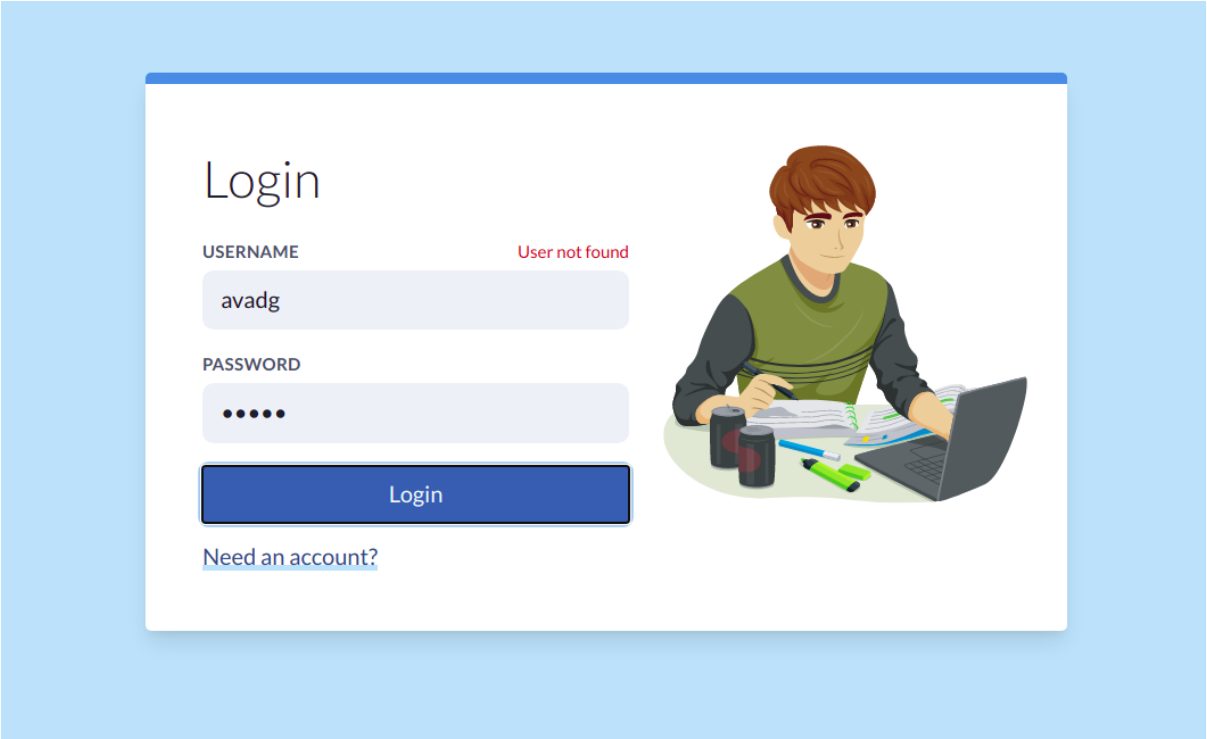
8. Implementation

Login Page

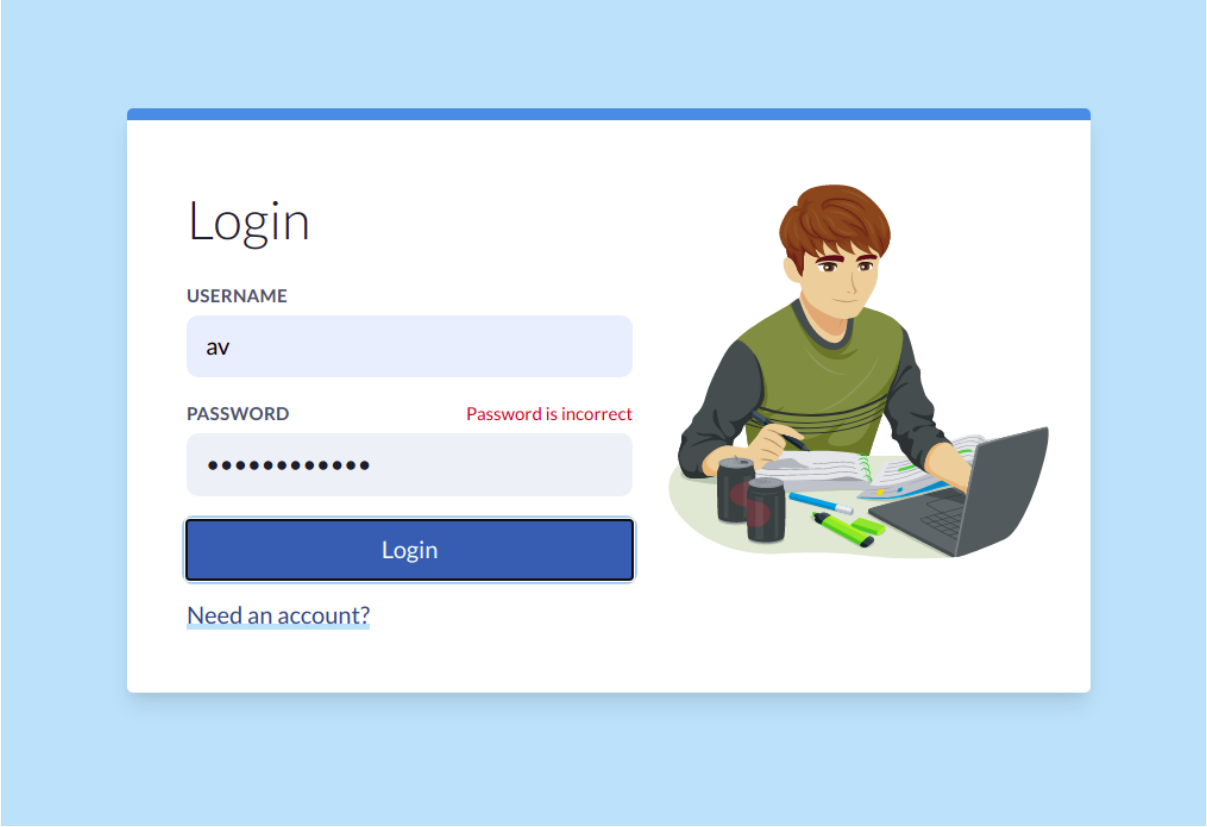


form validation

username checker



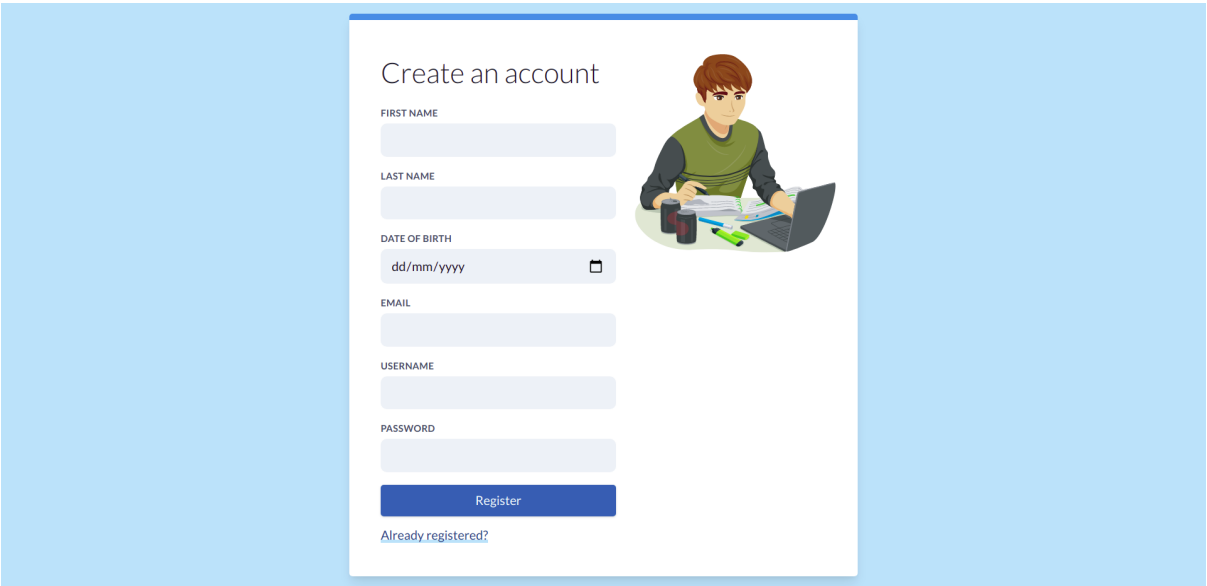
password checker



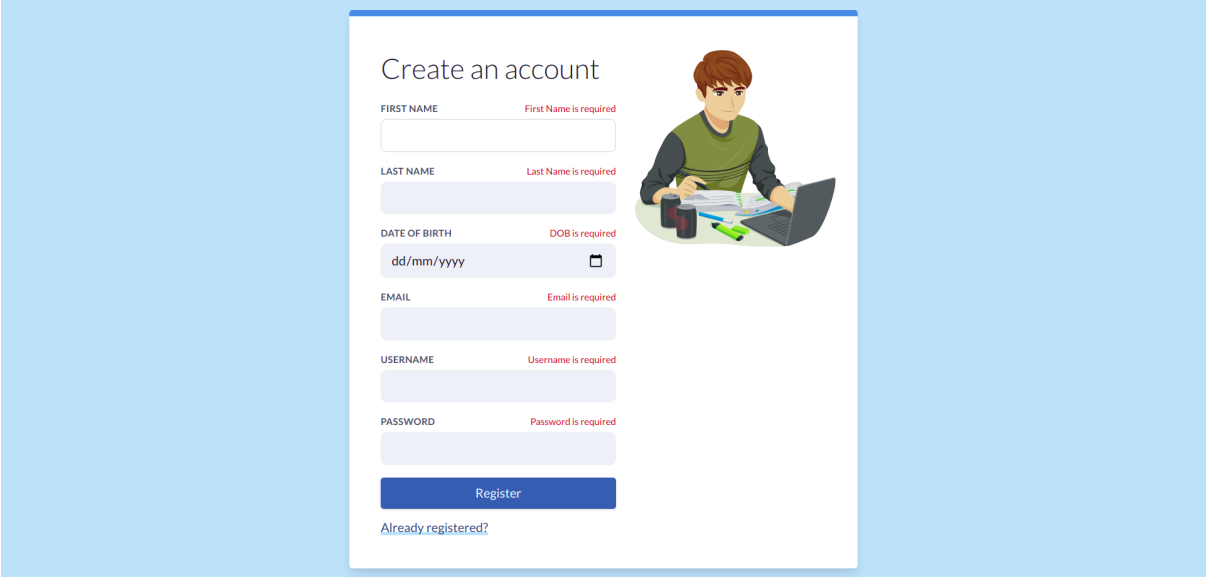
```
function Login() {
  const form = useForm();
  const history = useHistory();
  const [user, setUser] = React.useContext(UserContext);

  useEffectOnce(() => {
    if (user) {
      history.replace('/');
    }
  });

  const onSubmit = React.useCallback(
    formData =>
      axios
        .post('/api/users/auth', formData)
        .then(response => response.data)
        .then(data => setUser({ ...data.user, token: data.token }))
        .then(() => history.push('/chat'))
        .catch(handleRequestValidationError(form)),
    [form, history, setUser],
  );

```

```jsx
  return (
    <div className="flex justify-center items-center h-screen">
      <FormContext {...form}>
        <UserCardForm
          header="Login"
          image={StudentGraphic}
          alt="Boy Studying"
          underButton={<FancyLink to="/register">Need an account?</FancyLink>}
          onSubmit={form.handleSubmit(onSubmit)}
        >
          <UserCardFormField
            label="Username"
            name="username"
            type="text"
            register={form.register({ required: 'Username is required' })}
          />
          <UserCardFormField
            label="Password"
            name="password"
            type="password"
            register={form.register({ required: 'Password is required' })}
          />
        </UserCardForm>
      </FormContext>
    </div>
  );
}

export default Login;
```

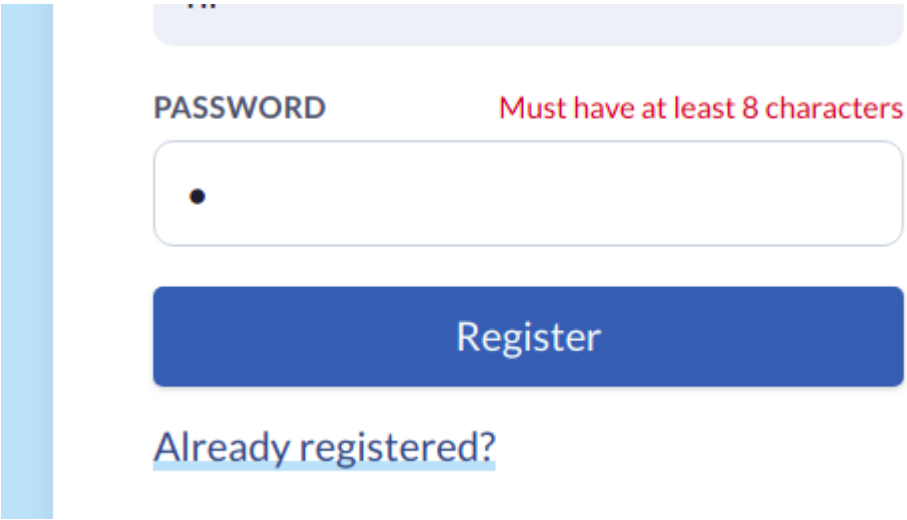
Register Page



validation of input



password parameter checking



```
function Register() {
  const [registered, setRegistered] = React.useState(false);
  const form = useForm();
  const history = useHistory();
  const [user] = React.useContext(UserContext);

  useEffectOnce(() => {
    if (user) {
      history.replace("/");
    }
  });

  const onSubmit = React.useCallback(
    (data) =>{
      console.log(data);
      axios
        .post("/api/users", data)
        .then(() => setRegistered(true))
        .catch(handleRequestValidationError(form)),
    [form]}
  );

  if (registered) {
    return (
      <div className="flex justify-center items-center h-screen">
      <Card className="flex flex-col items-center">
        <h1 className="text-4xl font-light mb-4">You&apos;re In!</h1>
        <img alt="Boy studying" src={StudentGraphic} style={{ width: 400 }} />
        <div className="text-2xl mt-8">
          <FancyLink to="/login">Login and start chatting!</FancyLink>
        </div>
      </Card>
    </div>
    );
  }
```

```
return (
  <div className="flex justify-center items-center h-screen">
    <FormContext { ...form}>
      <UserCardForm
        header="Create an account"
        image={StudentGraphic}
        alt="Student Studying"
        buttonText="Register"
        underButton={<FancyLink to="/login">Already registered?</FancyLink>}
        onSubmit={form.handleSubmit(onSubmit)}
      >
        <UserCardFormField
          label="First Name"
          name="firstname"
          type="text"
          register={form.register({
            required: "First Name is required",
            // validate: isEmpty
          })}
        />
        <UserCardFormField
          label="Last Name"
          name="lastname"
          type="text"
          register={form.register({
            required: "Last Name is required",
            // validate: isEmpty
          })}
        />
        <UserCardFormField
          label="Date of Birth"
          name="dob"
          type="date"
          register={form.register({
            required: "DOB is required",
            validate: isDate,
          })}
        />
```

```
            <UserCardFormField
              label="Email"
              name="email"
              type="email"
              register={form.register({
                required: "Email is required",
                validate: R.o(R.or(R.__, "Not a valid Email"), isEmail),
              })}
            />

            <UserCardFormField
              label="Username"
              name="username"
              type="text"
              register={form.register({ required: "Username is required" })}
            />
            <UserCardFormField
              label="Password"
              name="password"
              type="password"
              register={form.register({
                required: "Password is required",
                minLength: {
                  value: 8 ,
                  message: "Must have at least 8 characters",
                },
              })}
            />
          </UserCardForm>
        </FormContext>
      </div>
    );
  }

  export default Register;
```

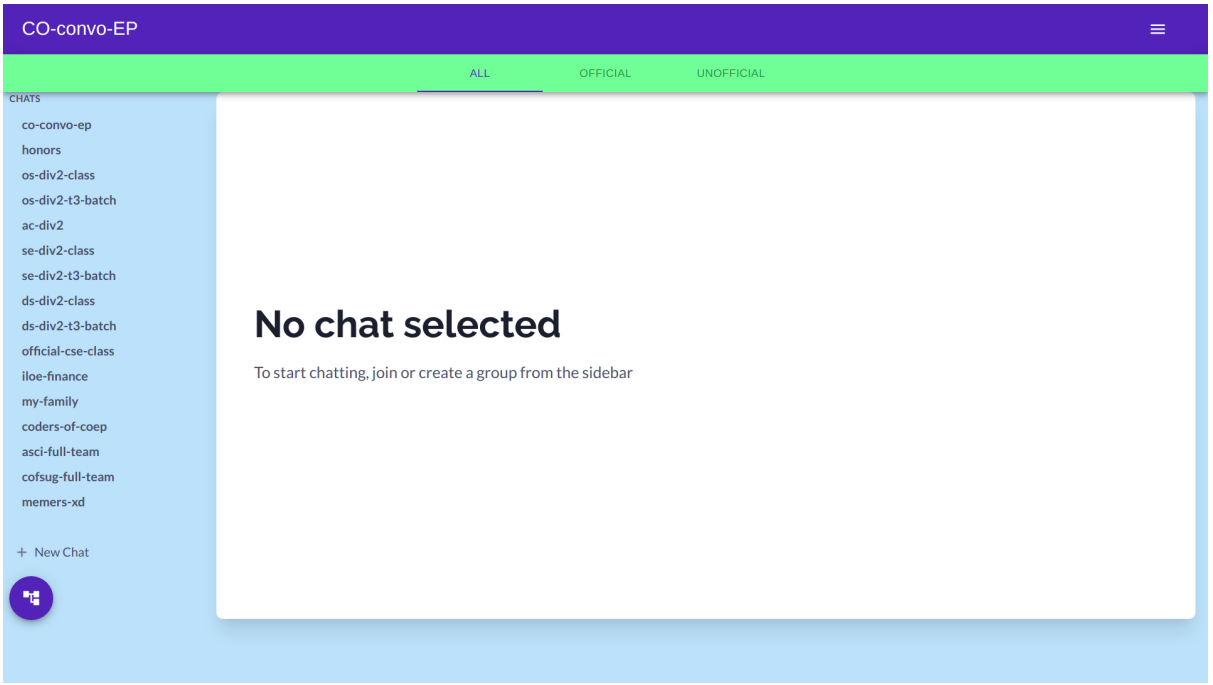
Home Page

without persistent connection

with persistent connection

```jsx
function Home() {
  return (
    <div className="relative mb-24">
      <div className="absolute z-0 w-32 mt-64 h-64" style={dottedBackground} />
      <div
        className="absolute z-0 w-8 mt-16 h-screen right-0"
        style={dottedBackground}
      />
      <div className="relative z-10 container mx-auto px-4">
        <div className="flex flex-wrap justify-center lg:pt-24">
          <div className="lg:w-1/2 mt-16 flex flex-col items-center lg:items-start">
            <h1 className="font-extrabold text-5xl leading-tight font-display text-center lg:text-left">
              CO {" "}
              <span className="text-blue-700">convo</span>
              {" "} EP
            </h1>
            <p className="text-black-700 text-xl mt-2 text-center lg:text-left">
              ChatApp for Colleges.
            </p>
            <nav className="mt-6 whitespace-no-wrap">
              <HomeActionButtons />
            </nav>
          </div>
          <div className="lg:w-1/2 mt-12">
            <img
              alt="CO convo EP logo"
              src={Logo}
              className="px-8 md:px-24 lg:px-0"
            />
          </div>
        </div>
      </div>
    </div>
  );
}

export default Home;
```

Chat Page


CO-convo-EP
ALL
OFFICIAL
UNOFFICIAL
CHATS
co-convo-ep
honors
os-div2-class
os-div2-t3-batch
ac-div2
se-div2-class
se-div2-t3-batch
ds-div2-class
ds-div2-t3-batch
official-cse-class
iloe-finance
my-family
coders-of-coep
asci-full-team
cofsug-full-team
memers-xd
New Chat
No chat selected
To start chatting, join or create a group from the sidebar

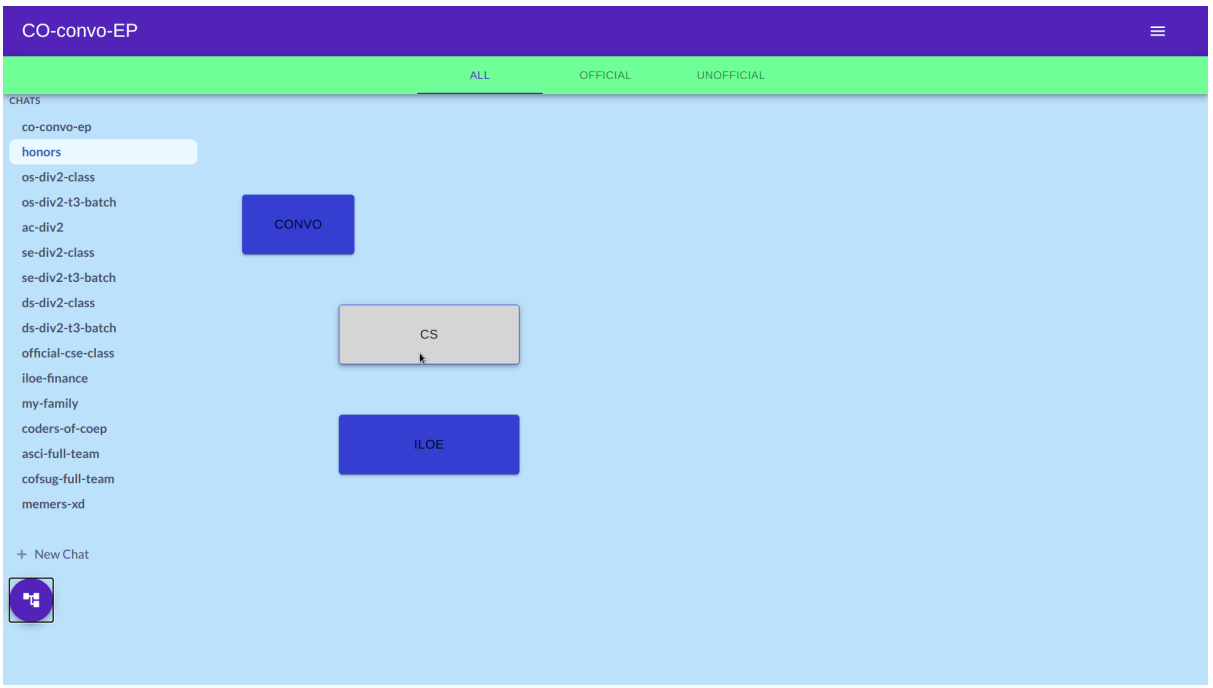
```
function Chat() {
  const [{ state, performEffect }, dispatch] = React.useReducer(
    chatReducer,
    undefined,
    init
  );
  const history = useHistory();
  const [user] = React.useContext(UserContext);

  // subscribe to socket emit events
  useEffectOnce(() => {
    const { socket } = state;

    // handle chat messages
    socket.on("chat message", (chatMessage) =>
      dispatch({ type: "recieve-message", payload: { chatMessage } })
    );

    // handle "user is typing"
    socket.on("start typing", (username) =>
      dispatch({ type: "add-user-typing", payload: { username } })
    );

    // remove "user is typing"
    socket.on("stop typing", (username) =>
      dispatch({ type: "remove-user-typing", payload: { username } })
    );

    return socket.close.bind(socket);
  });

  // go to login page if user is logged out
  useEffectOnce(() => {
    if (!user) {
      history.replace("/login");
    }
  }, [history, user]);

```

```
  React.useEffect(() => {
    if (performEffect) {
      performEffect(dispatch);
    }
  }, [dispatch, performEffect]);

  const chatContextValue = React.useMemo(() => [state, dispatch], [state]);

  const [treeViewVisible, setTreeViewVisible] = useState(false);

  const [groupType, setGroupType] = useState(0);

  return (
    <div>
      <AppBarView groupType={groupType} setGroupType={setGroupType} />
      <ChatContext.Provider value={chatContextValue}>
        <div className="flex">
          <ChatChannelSidebar channels={state.channels} groupType={groupType} />

          <div className="h-screen-85 mt-20 mr-4 flex flex-grow">
            {treeViewVisible ? (
              <TreeView
                setTreeViewVisible={setTreeViewVisible}
                channels={state.channels}
                groupType={groupType}
              />
            ) : (
              <ChatContent />
            )}
          </div>
        </div>
        <FAB
          treeViewVisible={treeViewVisible}
          setTreeViewVisible={setTreeViewVisible}
        />
      </ChatContext.Provider>
    </div>
  );
}
```

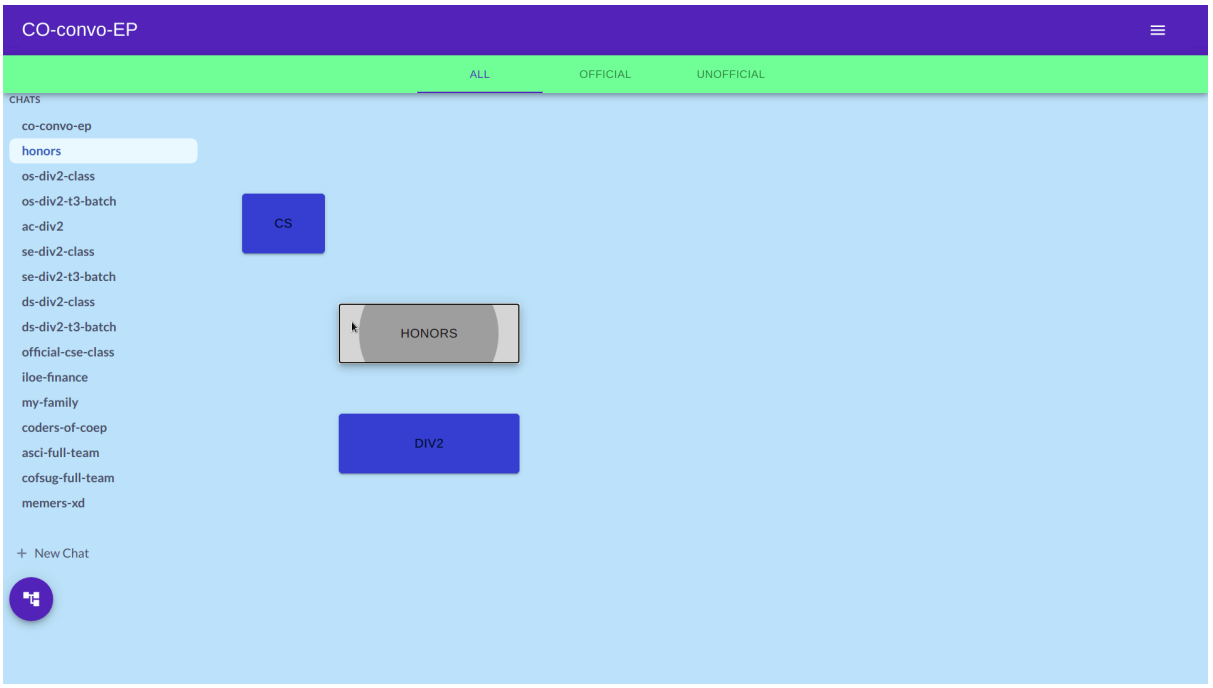
Tree View

tree view page

tree view stage 1



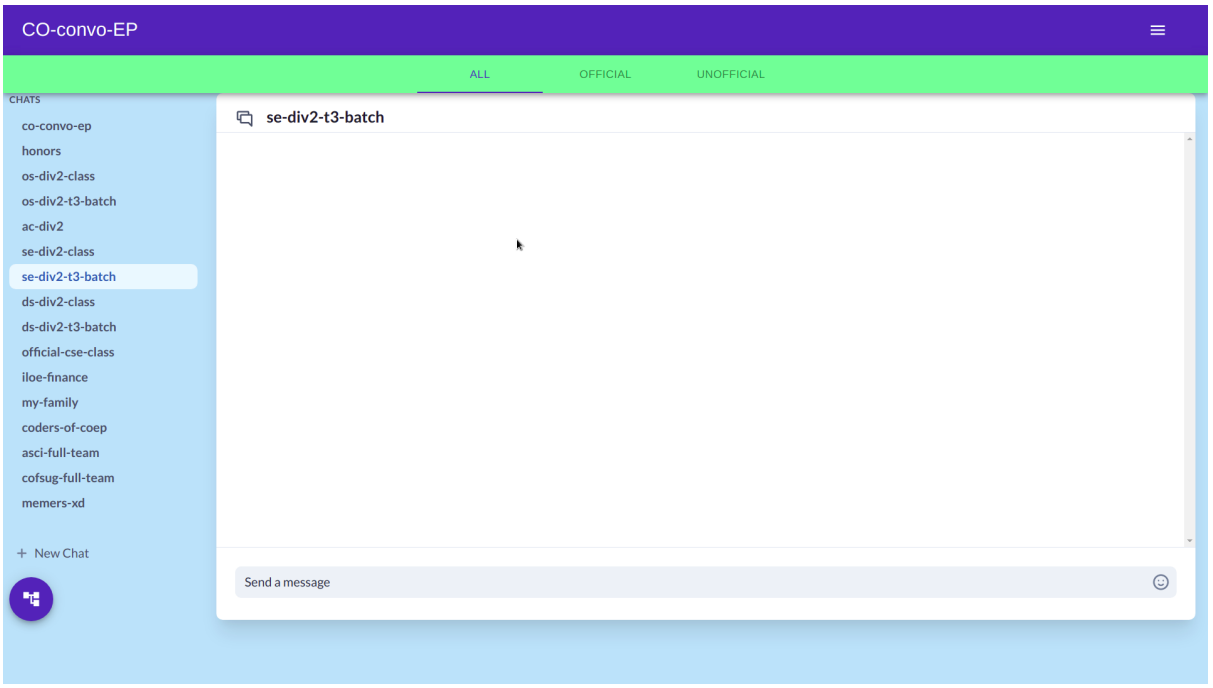
tree view stage 2



tree view stage 3

tree view stage 4

tree view leaf group clicked

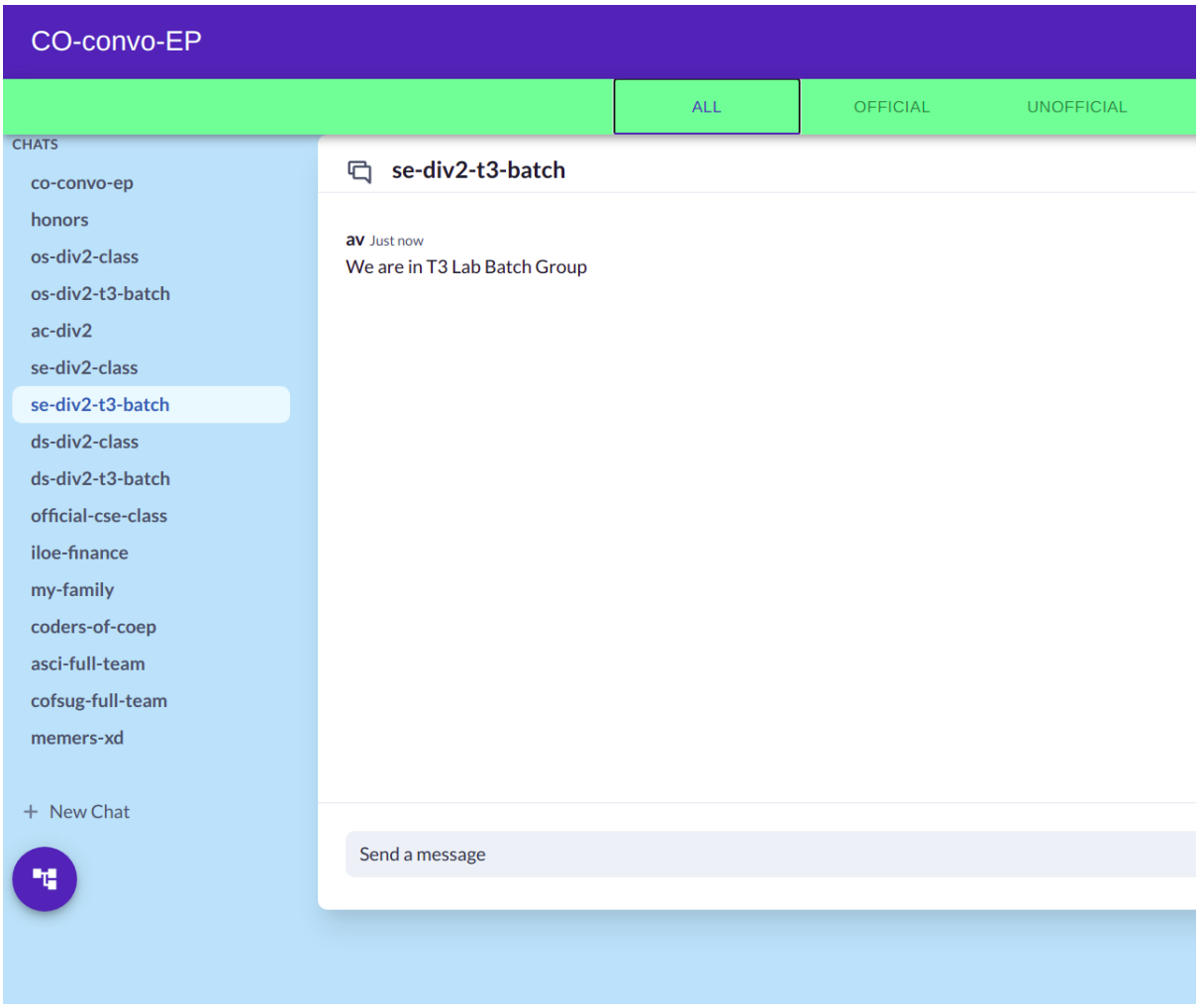


```
const TreeView = ({ channels, groupType, setTreeViewVisible }) => {
  const [state, dispatch] = React.useContext(ChatContext);
  const [user] = React.useContext(UserContext);
  const [parent, setParent] = useState(TreeItems);

  const changeChannel = React.useCallback(
    () => dispatch({ type: "change-channel", payload: { channels, user } }),
    [dispatch, user, channels]
  );

  const [backTrackArray, updateBackTrackArray] = useState([]);

  const onChildClick = (parent, child) => {
    if (child["children"].length !== 0) {
      backTrackArray.push(parent);
      updateBackTrackArray(backTrackArray);
      setParent(child);
    } else {
      // console.log("Open Chatbox", child.id);
      const channel = channels.find((temp) => temp._id === child.id);
      setTreeViewVisible(false);
      // console.log(channel);
      dispatch({ type: "change-channel", payload: { channel, user } }),
        [dispatch, user, channel];
      // changeChannel(channel);
      // state.fetchingChannel ? R.always() : changeChannel;
    }
  };

  const onParentClick = () => {
    if (backTrackArray.length !== 0) {
      setParent(backTrackArray.pop());
    }
  };

  const classes = useStyles();
```

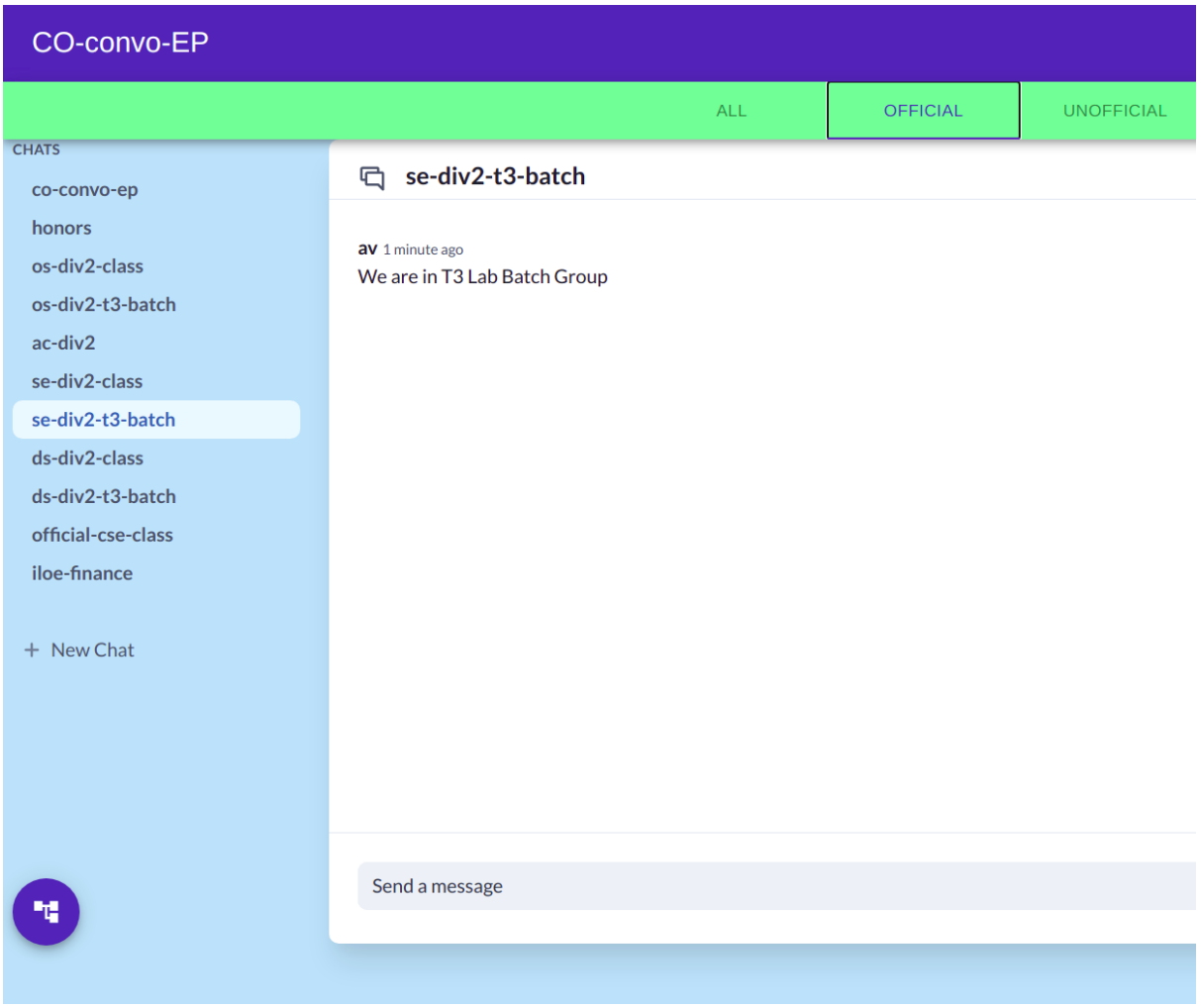
```
  return (
    <div className={classes.treeViewContainer}>
      <Button
        variant="contained"
        className={classes.parentButton}
        onClick={() => onParentClick()}
      >
        {parent["blockName"]}
      </Button>
      {parent["children"].map((children) => (
        <div>
          <div className="children">
            <Button
              variant="contained"
              className={classes.childButton}
              onClick={() => onChildClick(parent, children)}
            >
              {children["blockName"]}
            </Button>
          </div>
        </div>
      ))}
    </div>
  );
};

TreeView.propTypes = {
  channels: PropTypes.shape({
    _id: PropTypes.string,
    name: PropTypes.string,
    groupType: PropTypes.number,
  }).isRequired,
};

export default React.memo(TreeView);
```

Group Types Filtering

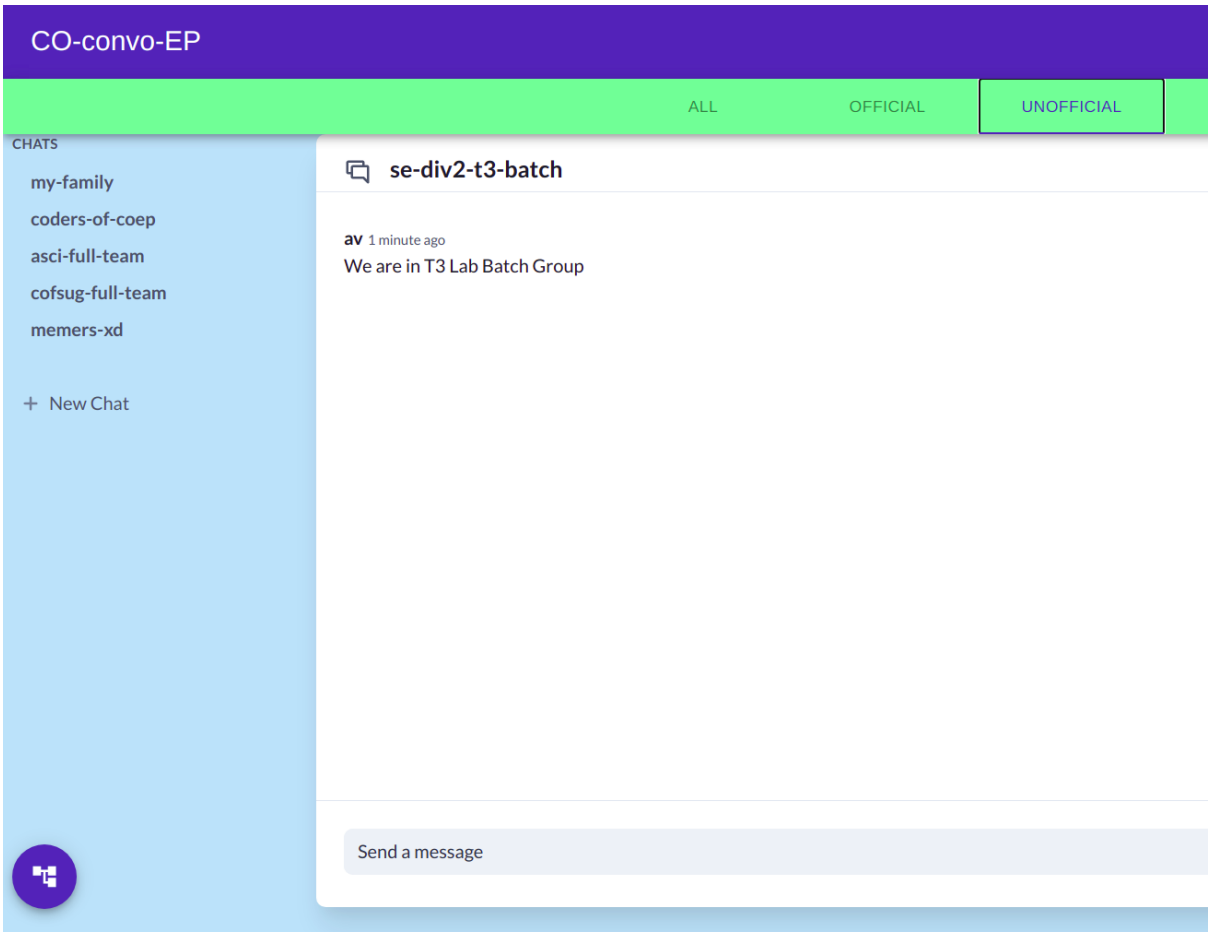
All Groups visible



official groups visible



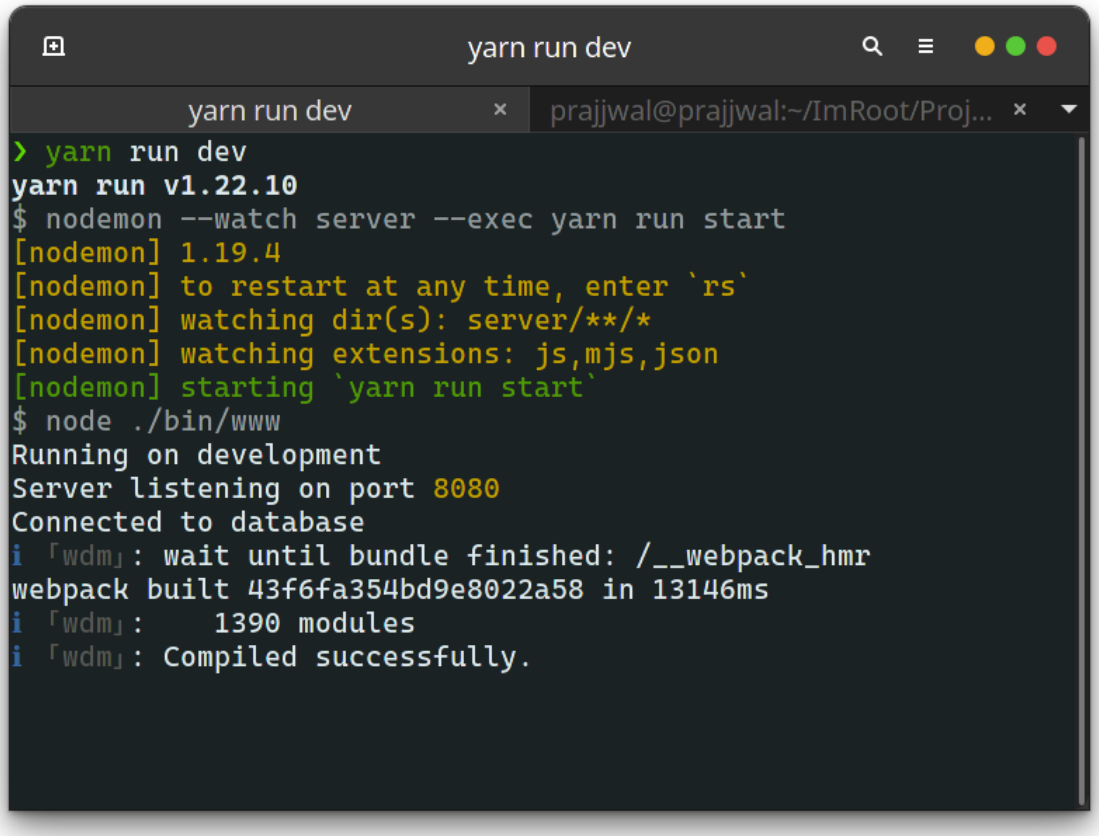
unofficial groups visible



```
const useStyles = makeStyles({
  root: {
    flexGrow: 1,
    position: "fixed",
    width: "100%",
    backgroundColor: "rgba(155,255,155,1)",
    borderRadius: 0,
  },
});

const TabView = ({ groupType, setGroupType }) => {
  const classes = useStyles();
  const [value, setValue] = React.useState(0);
  const handleChange = (event, newValue) => {
    setValue(newValue);
  };
  useEffect(() => {
    setGroupType(value);
  }, [value]);
  return (
    <Paper className={classes.root} elevation={5}>
      <Tabs
        value={value}
        onChange={handleChange}
        indicatorColor="primary"
        textColor="primary"
        centered
      >
        <Tab label="ALL" value={0} />
        <Tab label="OFFICIAL" value={1} />
        <Tab label="UNOFFICIAL" value={2} />
      </Tabs>
    </Paper>
  );
};
export { TabView };
```

```
  return groupType === 0 ? (
    <button
      className={`block w-full text-left p-1 pl-4 rounded-lg truncate font-bold ${
        state.currentChannel && state.currentChannel._id === channel._id
          ? "text-blue-700 hover:text-blue-900 bg-blue-100"
          : "hover:text-gray-900 hover:bg-gray-200"
      }`}
      type="button"
      onClick={state.fetchingChannel ? R.always() : changeChannel}
    >
      {channel.name}
    </button>
  ) : groupType === 1 && channel.groupType === 1 ? (
    <button
      className={`block w-full text-left p-1 pl-4 rounded-lg truncate font-bold ${
        state.currentChannel && state.currentChannel._id === channel._id
          ? "text-blue-700 hover:text-blue-900 bg-blue-100"
          : "hover:text-gray-900 hover:bg-gray-200"
      }`}
      type="button"
      onClick={state.fetchingChannel ? R.always() : changeChannel}
    >
      {channel.name}
    </button>
  ) : groupType === 2 && channel.groupType === 2 ? (
    <button
      className={`block w-full text-left p-1 pl-4 rounded-lg truncate font-bold ${
        state.currentChannel && state.currentChannel._id === channel._id
          ? "text-blue-700 hover:text-blue-900 bg-blue-100"
          : "hover:text-gray-900 hover:bg-gray-200"
      }`}
      type="button"
      onClick={state.fetchingChannel ? R.always() : changeChannel}
    >
      {channel.name}
    </button>
  ) : (
    <></>
  );
}
```

Running the project on terminal



9. Testing

9.1 Black Box Testing

Black Box Testing for Login Module

MODULE NAME	LOGIN											
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRECONDITIONS	TEST DATA	POST CONDITIONS	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTED BY	EXECUTED DATE(MM-DD-YYYY)
TS_BN_001	Verify the Login functionality of Login page	TC_BN_Login_001	Enter Valid Username and Password	1. Enter Valid Username 2. Enter Valid Password 3. Click on Login Button	Valid URL	Username : convo Password : 12345678	User should be redirected to homepage	Logged in successfully	Logged in successfully	Pass	Team CO-convo-EP	04-24-2021
TS_BN_001	Verify the Login functionality of Login page with Invalid Username	TC_BN_Login_002	Enter Invalid Username and Password	1. Enter Invalid Username 2. Enter Valid Password 3. Click on Login Button	Valid URL	Username : ****** Password : 123456	Invalid username or password	Error message is shown	Error message is shown	Pass	Team CO-convo-EP	04-24-2021
TS_BN_001	Verify the Login functionality of Login page with Invalid Password	TC_BN_Login_003	Enter valid username and invalid password	1. Enter Valid Username 2. Enter Invalid Password 3. Click on Login Button	Valid URL	Username: convo Password: 1234	Invalid username or password	Error message is shown	Error message is shown	Pass	Team CO-convo-EP	04-24-2021
TS_BN_001	Verify the Login functionality of Login page with Invalid username and Password	TC_BN_Login_004	Enter invalid username and invalid password	1. Enter Valid Username 2. Enter Invalid Password 3. Click on Login Button	Valid URL	Username: ****** Password: *******	Invalid username or password	Error message is shown	Error message is shown	Pass	Team CO-convo-EP	04-24-2021

Black Box Testing for Register module

MODULE NAME :	REGISTER											
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRECONDITIONS	TEST DATA	POST CONDITIONS	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTED BY	EXECUTED DATE(MM-DD-YYYY)
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_001	Enter valid available username, valid email, valid password and other fields [FirstName, LastName, DOB]	1. Enter available username 2. Enter valid email 3. Enter valid password 4.Enter Other Fields	Valid URL	Username : Aditimedhane Email : medhaneaditi@gmail.com Password : ******** FirstName : Aditi Last Name : Medhane DOB : 15/06/2001	User Should be redirected to Login page	Registration Successful	Registration Successful	Pass	Team CO-convo-EP	04-24-2021
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_002	Enter valid email, valid password and other fields [FirstName, LastName, DOB] But Unavailable Username	1. Enter unavailable username 2. Enter valid email 3. Enter valid password 4.Enter Other Fields	Valid URL	Username : Aditimedhane Email : medhanear18.comp@coep.ac Password : ******** FirstName : Aditi Last Name : Medhane DOB : 15/06/2001	User should be prompted with error	Error Message is shown	Error Message is shown	Pass	Team CO-convo-EP	04-24-2021
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_003	Enter valid Username, valid password and other fields [FirstName, LastName, DOB] But Unavailable Email address	1. Enter unavailable email 2. Enter valid username 3. Enter valid password 4.Enter Other Fields	Valid URL	Username : medhaneaditi Email : medhanear18.comp@coep.ac Password : ******** FirstName : Aditi Last Name : Medhane DOB : 15/06/2001	User should be prompted with error	Error Message is shown	Error Message is shown	Pass	Team CO-convo-EP	04-24-2021
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_004	Enter valid email, valid Username and other fields [FirstName, LastName, DOB] But Invalid Password.	1. Enter invalid password 2. Enter valid email 3. Enter valid password 4.Enter Other Fields	Valid URL	Username : prajjwal Email : datirpr18.comp@coep.ac.in Password : ******** FirstName : Prajjwal Last Name : Datir DOB : 15/06/2001	User should be prompted with error	Error Message is shown	Error Message is shown	Pass	Team CO-convo-EP	04-24-2021
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_005	Enter valid username, valid password and other fields [FirstName, LastName, DOB] But Invalid Email address	1. Enter invalid email 2. Enter valid username 3. Enter valid password 4.Enter Other Fields	Valid URL	Username : GaneshGitte Email : Ganesh18 Password : ******** FirstName : Ganesh Last Name : Gitte DOB : 15/06/2001	User should be prompted with error	Error Message is shown	Error Message is shown	Pass	Team CO-convo-EP	04-24-2021

Black Box testing for Chat Module

MODULE NAME :	CHAT											
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRECONDITIONS	TEST DATA	POST CONDITIONS	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTED BY	EXECUTED DATE(MM-DD-YYYY)
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_001	Type Message	Type a message in the text box	User Logged in	msgData : hello	"typing" should be true	Message visible in the text box	Message visible in the text box	Pass	Team CO-convo-EP	04-24-2021
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_002	Send Message	repeat steps of TC_BN_Chat_001 and Click on Send button	User Logged in	clickButton send	message should be sent to the server	Message is sent	Message is sent	Pass	Team CO-convo-EP	04-24-2021
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_003	Recieve Message	repeat steps of TC_BN_Chat_002 and Monitor UI changes	User Logged in	sendMessage from TD_BN_Chat_002	message should be recieve to the server	Message is recieved	Message is recieved	Pass	Team CO-convo-EP	04-24-2021
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_004	Message author	repeat steps of TC_BN_Chat_003 and inspect message author	User Logged in	sendMessage from TD_BN_Chat_002	message author should be visible	Author visible	Author visible		Team CO-convo-EP	

Black Box testing for Tree view module

MODULE NAME :	TREE VIEW											
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRECONDITIONS	TEST DATA	POST CONDITIONS	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTED BY	EXECUTED DATE(MM-DD-YYYY)
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_00	Check if Floating Action Button exists	inspect for floating action button	User Logged in	should be on the homepage	floating window shall appear	floating action button visible is true	floating action button visible is true	Pass	Team CO-convo-EP	04-25-2021
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_00	check if front navigation is working	click on front navigation	User Logged in	floating action button should be on	Tree should have 1 level dows nondes	Go one level down	Go one level down	Pass	Team CO-convo-EP	04-25-2021
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_00	check if back navigation is working	click on back navigation	User Logged in	should be 1 level inside tree	Tree should have 1 level up node	Come one level up	Come one level up	Pass	Team CO-convo-EP	04-25-2021
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_00	check if tree leaves redirect to chat window	click on tree leaves	User Logged in	should be on the tree leave	It should be the tree leave	Open the chat window	Open the chat window	Pass	Team CO-convo-EP	04-25-2021

9.2 White Box Testing

White Box testing for Login and Register module

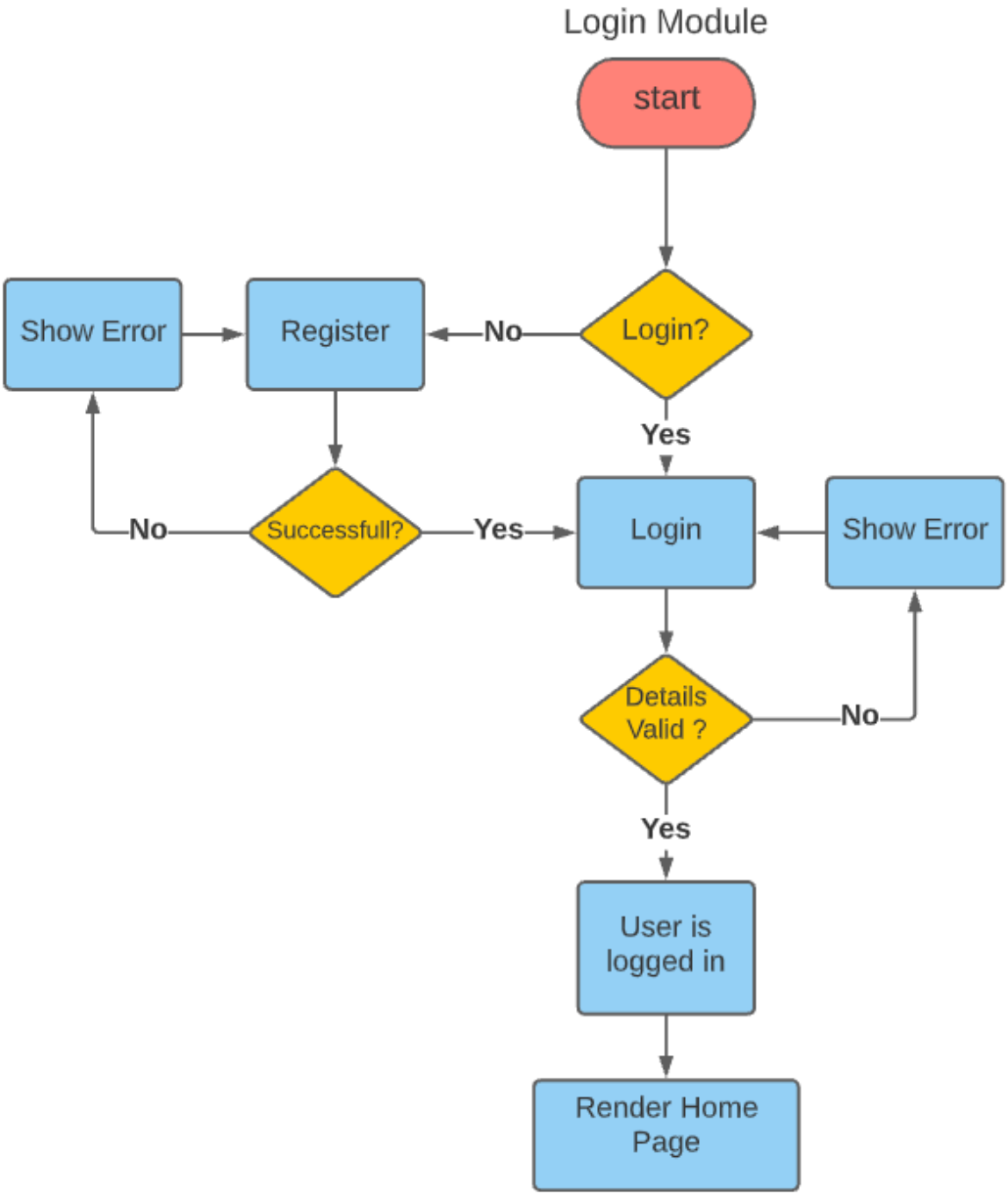
MODULE NAME	LOGIN							
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	PRECONDITIONS	PATH	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS
TS_BN_001	Verify the Login functionality of Login page	TC_BN_Login_001	Valid URL	1,5,6,8,9	Enter Valid Username and Password	Logged in successfully	Logged in successfully	Pass
TS_BN_001	Verify the Login functionality of Login page with Invalid Username	TC_BN_Login_002	Valid URL	1,5,6,7,5,6,8,9	Enter Invalid Username and Password	Error message is shown	Error message is shown	Pass
TS_BN_001	Verify the Login functionality of Login page with Invalid Password	TC_BN_Login_003	Valid URL	1,5,6,7,5,6,8,9	Enter valid username and invalid password	Error message is shown	Error message is shown	Pass
TS_BN_001	Verify the Login functionality of Login page with Invalid username and Password	TC_BN_Login_004	Valid URL	1,5,6,7,5,6,8,9	Enter invalid username and invalid password	Error message is shown	Error message is shown	Pass

Algorithm

MODULE NAME :	REGISTER							
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	PRECONDITIONS	PATH	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_001	Valid URL	1,2,3,5,6,8,9	Enter valid available username, valid email, valid password and other fields [FirstName, LastName, DOB]	Registration Successful	Registration Successful	Pass
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_002	Valid URL	1,2,3,4,2,3,5,6,8,9	Enter valid email, valid password and other fields [FirstName, LastName, DOB] But Unavailable Username	Error Message is shown	Error Message is shown	Pass
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_003	Valid URL	1,2,3,4,2,3,5,6,8,9	Enter valid Username, valid password and other fields [FirstName, LastName, DOB] But Unavailable Email address	Error Message is shown	Error Message is shown	Pass
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_004	Valid URL	1,2,3,4,2,3,5,6,8,9	Enter valid email, valid Username and other fields [FirstName, LastName, DOB] But Invalid Password.	Error Message is shown	Error Message is shown	Pass
TD_BN_002	Verify the register/signup functionality of Signup Page	TC_BN_Signup_005	Valid URL	1,2,3,4,2,3,5,6,8,9	Enter valid username, valid password and other fields [FirstName, LastName, DOB] But Invalid Email address	Error Message is shown	Error Message is shown	Pass

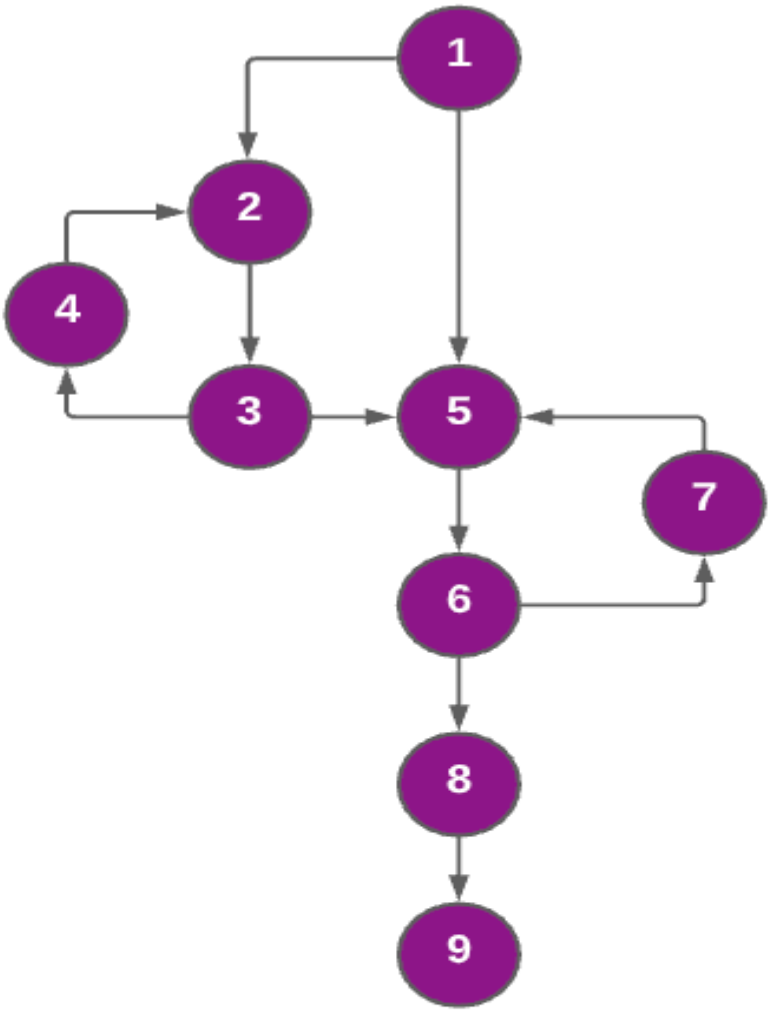
If login selected :
    Go to login page
    Take user credentials
    If credentials are valid :
        Login user
        Render Home page
    Else :
        Show error message
        Redirect to login page again
Else :
    Go to register page
    Take the user input
    If Registration is successful :
        Go to login page for logging in the user
    Else :
        Show error message
        Redirect to register page again

Flowchart



Flowchart

Flowgraph



Flowgraph

Cyclomatic complexity analysis = E-V+2
= 11 - 9 + 2
= 4

Paths :

1, 5, 6, 8, 9
1, 5, 6, 7, 5, 6, 8, 9
1, 2, 3, 5, 6 , 8, 9
1, 2, 3, 4, 2, 3, 5, 6, 8, 9

White box testing for Chat module

MODULE NAME :	CHAT							
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	PRECONDITIONS	PATH	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_001	User Logged in	1,2,1,2,3,4,9	Check if User selected contact or group for chatting	With no group/contact selected view available groups	With no group/contact selected view available groups	Pass
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_002	User Logged in	1,2,3,4,9	Check if User is sending the message	Without input message, view chat history	Without input message, view chat history	Pass
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_003	User Logged in	1,2,3,4,5,6,7,9	Check if User is sending the message	With input message, send the message to all group members	With input message, send the message to all group members	Pass
TD_BN_003	Verify chat functionality of chat page	TC_BN_Chat_004	User Logged in	1,2,3,4,5,6,8,6,7,9	Check if Message is sent	If message is not sent, waiting for message to get sent and view chat history after that	If message is not sent, waiting for message to get sent and view chat history after that	Pass

Algorithm

Do

View available groups/contacts

If a group or contact is selected :

Load messages of respective group/contact

If user sends a message :

Send message to selected group/contact

If message is sent :

Add message to chat history

Show chat history

Else :

Wait for message to get sent

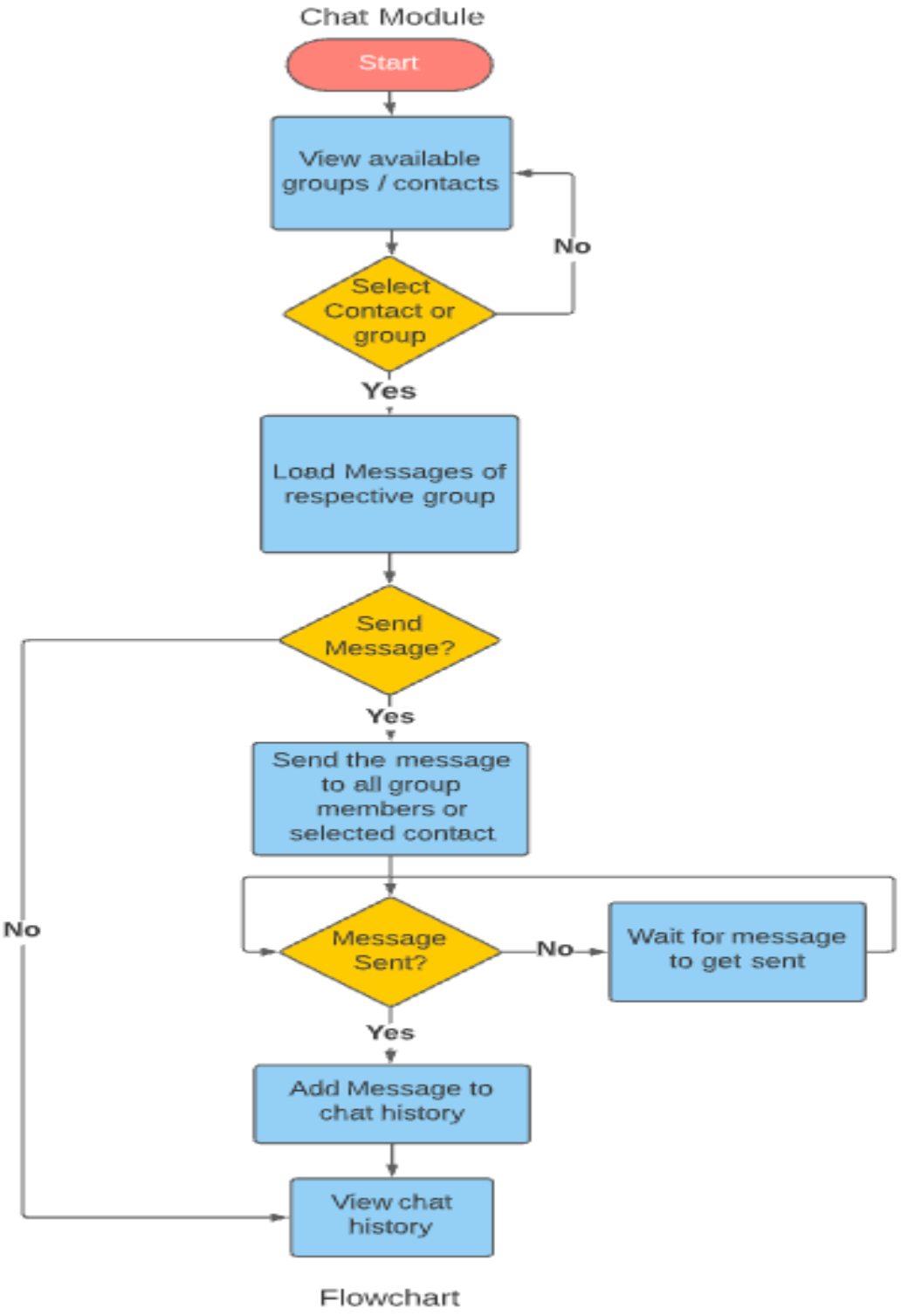
Else :

Show chat history

Else :

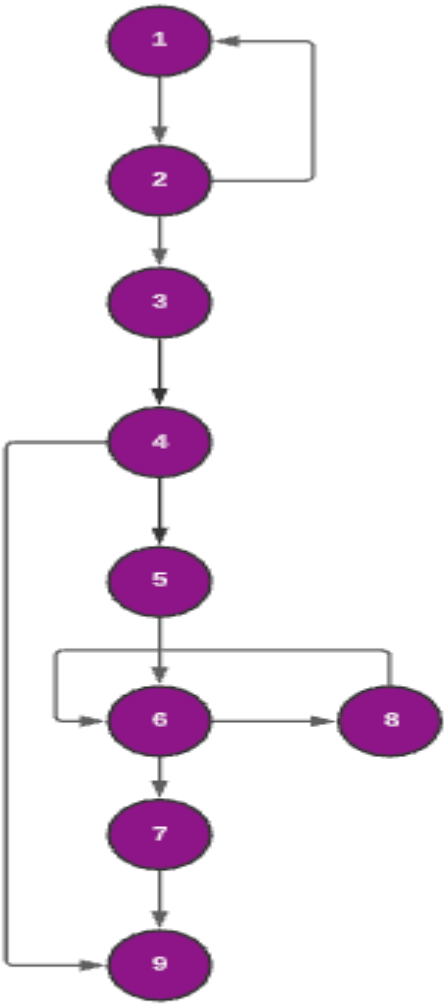
Show available groups/contacts

Flowchart



Flowchart

Flowgraph



Flowgraph

Cyclomatic complexiity = E - V + 2
= 11 - 9 + 2
= 4

Paths :

1, 2, 1, 2, 3, 4, 9
1, 2, 3, 4, 9
1, 2, 3, 4, 5, 6, 7, 9
1, 2, 3, 4, 5, 6, 8, 6, 7, 9

White Box testing for Tree View Module

MODULE NAME :	TREE VIEW							
TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	PRECONDITIONS	PATH	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_001	User Logged in	1,2,1,2,3,4,5	Check if the user has opened the tree view	If tree view not selected, show group list view	If tree view not selected, show group list view	Pass
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_002	User Logged in	1,2,3,4,3,4,5	Check if the user reached the desired group	If not reached the desired group, then keep navigating	If not reached the desired group, then keep navigating	Pass
TD_BN_004	Verify the tree view feature of tree view page	TC_BN_Tree_View_003	User Logged in	1,2,3,4,5	Check if the user reached the desired group	If reached the desired group, open chatbox of respective group	If reached the desired group, open chatbox of respective group	Pass

Algorithm

Do

View list of groups/contacts

If tree view selected:

Navigate through branches of tree view

If desired group is reached:

Open chat box of respective group

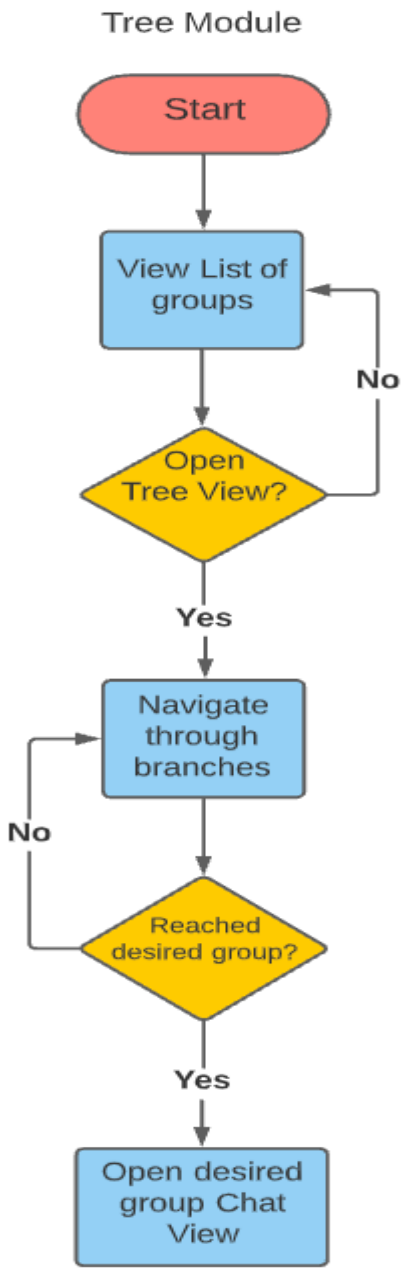
Else:

Keep navigating through branches

Else:

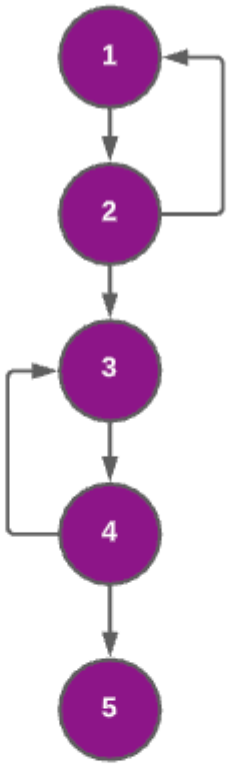
View list of groups/contacts

Flowchart



Flowchart

Flowgraph



Flowgraph

Cyclomatic complexiity = E - V + 2
= 6 - 5 + 2
= 3

Paths :

1,2,1,2,3,4,5
1,2,3,4,3,4,5
1,2,3,4,5

10.Gitlab Link

https://gitlab.com/TheCodingWizard-911/SE-Project

11. References

1) https://www.researchgate.net/publication/322509087_Developing_an_End-to-End_Secure_Chat_Application

2) https://www.researchgate.net/publication/234793505_Design_of_a_secure_chat_application_based_on_AES_cryptographic_algorithm_and_key_management

3) http://www.cse.tkk.fi/en/publications/B/10/papers/Vestola_final.pdf

4) https://projects.cecs.pdx.edu/attachments/download/7439/RHorrace-RequirementsDoc.pdf

5) https://repositori.udl.cat/bitstream/handle/10459.1/60232/groviras.pdf?sequence=1&isAllowed=y

6) https://www.slideshare.net/CrGaurav/a-project-report-on-chat-application

7) https://www.semanticscholar.org/paper/Security-analysis-of-end-to-end-encryption-in-Lee-Choi/93fe3a5e70d64964e775ea77dcfaee218b8e62e1

8) https://eprint.iacr.org/2016/1013.pdf

9) https://cornerstone.lib.mnsu.edu/cgi/viewcontent.cgi?article=1058&context=jur

10) https://www.researchgate.net/publication/316892755_Survey_Analysis_on_the_usage_and_Impact_of_Whatsapp_Messenger

11) https://www.scitepress.org/Papers/2017/63537/63537.pdf

12) https://core.ac.uk/download/pdf/143481552.pdf

13) http://www.jetir.org/papers/JETIR2005110.pdf

14) https://www.researchgate.net/publication/3283066_A_study_of_Internet_instant_messaging_and_chat_protocols

15) https://www.westminsterpapers.org/article/id/274/

16) https://www.hindawi.com/journals/misy/2018/2056290/