# COMSATS University Islamabad Abbottabad, Pakistan

Roomy Chat Rooms Platform

***By***

**M. Saad Hussain CIIT/SP21-BSE-020/ATD Muddassir Ali CIIT/SP21-BSE-016/ATD Muhammad Haris CIIT/SP21-BSE-019/ATD**

***Supervisor***

**Mr. Fuzel Jameel**

***Bachelor of Science in Computer Science (2021-2025)***

**The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others**.

.

# COMSATS University, Islamabad Abbottabad Campus

Roomy Chat Rooms Platform

## A project presented to

**COMSATS University Islamabad, Pakistan**

## In partial fulfillment

**of the requirement for the degree of**

***Bachelor of Science in Computer Science (2021-2025)***

## By

**M. Saad Hussain CIIT/SP21-BSE-020/ATD Muddassir Ali CIIT/SP21-BSE-016/ATD Muhammad Haris CIIT/SP21-BSE-019/ATD**

i

**DECLARATION**

We hereby declare that this software, neither whole nor as a part, has been copied out from any source. It is further declared that we have developed this software and accompanied the report entirely based on our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

**Muhammad Saad Husain Muddassir Ali Muhammad Haris**

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (SE) “Roomy Chat Rooms Platform” was developed by **Muhammad Saad Husain (CIIT/SP21-BSE-020), Muddassir Ali (CIIT/SP21-BSE-016), and Muhammad Haris (CIIT/SP21-BSE-019),** under the supervision of “Mr. Fuzel Jameel” and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

**Supervisor**

**External Examiner**

**Head of Department (Department of Computer Science)**

**EXECUTIVE SUMMARY**

Roomy is a web app powered by real-time communication in a chat room on different topics. Unlike other chat or messaging platforms, Roomy allows users to create temporary chat rooms on specific topics or interests and text or talk with others in these temporary rooms. The application uses MERN stack where we use MongoDB for data storage, Express.js in backend API development, React.js for frontend user interface, and Node. About Roomy, users can find and join rooms based on their interests, meet and interact with similar people, and take part in energetic discussions within an open and pleasurable space.

.

Key Features of Roomy include:

Room Creation and Discovery: Provide users with the ability to create new chat rooms by interest, hobby, or topic. Users can search rooms by their names and join rooms.

Real-time Chat: Support real time messaging exchanging through sockets adds functionalities like message threads, emojis, and reactions.

Audio Chat Rooms: Introduce audio chat rooms where users can engage with each other through audio conversation. Features which we use are joining, leaving, mute etc. for audio.

Room Scheduling: Users can schedule the room through calendar provide starting and ending date schedule room on that specific date room exists for meeting, discussions.

User Profiles and Preferences: Provide users to see their profiles and ability to change their names, profile images, bios, interests etc. and what they want.

Moderation and Administration: Implement moderation tools for room creators and administrators to manage chat room activities, follow community guidelines, and handle user reports

.

**ACKNOWLEDGEMENT**

All praise is due to Almighty Allah who granted us a small portion of his infinite wisdom, enabling us to successfully complete this demanding task.

We would like to extend our heartfelt gratitude to our project leader 'professor' “Mr. Fuzel Jameel”. absence from the project would have made its completion uncertain without his personal supervision, advice, and valuable guidance. We are immensely grateful to them for their unwavering support and constant assistance throughout this project.

And we are immensely grateful to our parents and family members who have consistently supported us and instilled in us the values of honesty and hard work.

**Muhammad Saad Husain Muddassir Ali Muhammad Haris**

-

**ABBREVIATIONS**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
| **FC** | Functional Requirements |
| **NFC** | Non-Functional Requirements |
| **UC** | Use Case |
| **DFD** | Data Flow Diagram |

**TABLE OF CONTENTS**

1. [Introduction 1](#_bookmark0)
   1. [Brief 1](#_bookmark1)
   2. [Relevance to Course Modules 1](#_bookmark2)
   3. [Project Background 2](#_bookmark3)
   4. [Literature Review 2](#_bookmark4)
   5. [Analysis from Literature Review (in the context of your project) 3](#_bookmark5)
   6. [Methodology and Software Lifecycle for This Project 3](#_bookmark6)
      1. [Rationale behind Selected Methodology 3](#_bookmark7)
2. [Problem Definition 5](#_bookmark8)
   1. [Problem Statement 5](#_bookmark9)
   2. [Deliverables and Development Requirements 5](#_bookmark10)
   3. [Current System (if applicable to your project) 6](#_bookmark11)
3. [Requirement Analysis 8](#_bookmark12)
   1. [Use Cases Diagram(s) 8](#_bookmark13)
   2. [Detailed Use Case 9](#_bookmark14)
   3. [Functional Requirements 20](#_bookmark15)
   4. [Non-Functional Requirements 21](#_bookmark16)
4. [Design and Architecture 23](#_bookmark17)
   1. [System Architecture 23](#_bookmark18)
   2. [Data Representation [Diagram + Description] 26](#_bookmark19)
   3. [Process Flow/Representation 36](#_bookmark20)
   4. [Design Models [along with descriptions] 37](#_bookmark21)
5. [Implementation 37](#_bookmark22)
   1. [Algorithm 39](#_bookmark23)
   2. [External APIs 42](#_bookmark24)
   3. [User Interface 42](#_bookmark25)
6. [Testing and Evaluation 54](#_bookmark26)
   1. [Manual Testing 54](#_bookmark27)
      1. [System testing 54](#_bookmark28)
      2. [Unit Testing 54](#_bookmark29)
      3. [Functional Testing 61](#_bookmark30)
      4. [Integration Testing 61](#_bookmark31)
   2. [Automated Testing: 66](#_bookmark32)
      1. [Tools used: 66](#_bookmark33)
7. [Conclusion and Future Work 67](#_bookmark34)
   1. [Conclusion 67](#_bookmark35)
   2. [Future Work 67](#_bookmark36)
8. [References 68](#_bookmark37)

**LIST OF FIGURES**

Fig 3.1: Use Case Diagram 8

Fig 4.1: Architecture diagram 25

Fig 4.2 System Sequance Diagram 26

Fig 4.3 Sequence diagram for Registration 27

Fig 4.4 Sequence diagram for Sign in 28

Fig 4.5 Sequence diagram for Manage Profile 29

Fig 4.6 Sequence diagram for create Chat Rooms 30

Fig 4.7 Sequence diagram for invite members 31

Fig 4.8 Sequence diagram for join chat room 32

Fig 4.9 Sequence diagram for search Rooms 33

Fig 4.10 Sequence diagram for update room 34

Fig 4.11 Sequence diagram for delete chat room 35

Fig 4.12 Process Flow/Representation 36

Fig 4.13 Data Flow Diagram level 0 37

Fig 4.14 Data Flow Diagram level 1 37

Fig 4.15 Data Flow Diagram level 2 38

Fig 5.1 Home Page 42

Fig 5.2 Profile page 43

Fig 5.3 Registration page 43

Fig 5.4 Login page 44

Fig 5.5 forget password screen 44

Fig 5.6 Reset password screen 45

Fig 5.7 messages page 45

Fig 5.8 List of Rooms page 46

Fig 5.9 List of users in group 47

Fig 5.10 Video call joining screen 48

Fig 5.11 Video calling screen 48

Fig 5.12 Screen sharing window 49

Fig 5.13 Messaging in video call screen 49

Fig 5.14 Private Room screen 50

Fig 5.15 Temporary Room Screen 50

Fig 5.16 Login Screen 51

Fig 5.17 Registration Screen 51

Fig 5.18 Forget password Screen 52

Fig 5.19 Create Room Screen 52

Fig 5.20 Profile Management Screen 53

Fig 5.21 Profile Management Screen 53

**LIST OF TABLES**

Table 3.1 Register 9

Table 3.2 Sign In 10

Table 3.3 Manage Profile 11

Table 3.4 Create Chat Room 12

Table 3.5 Invite Members 13

Table 3.6 Search Room 14

Table 3.7 Join Room 15

Table 3.8 Schedule Chat Room 16

Table 3.9 Update Chat Room 17

Table 3.10 Leave Room 17

Table 3.11 Delete chatrooms 18

Table 3.12 log Out 19

Table 5.1 Details of APIs used in the project 42

Table 6.1 System Testing Testcase 54

Table 6.2 Login Unit Testcase 56

Table 6.3 Profile Management Module Unit Testcase 57

Table 6.4 Chat Room Creation Module Test case 57

Table 6.5 temporary chat room creation Module Test case 58

Table 6.6 invitation module Test case 58

Table 6.7 search module Test case 59

Table 6.8 join chat rooms module Test case 59

Table 6.9 Room Scheduling module Test case 59

Table 6.10 Real time communication Test case 60

Table 6.11 Private rooms accessibility module Test case 60

Table 6.12 User Registration Functional Testcase 61

Table 6.13 User Login Functional Testcase 62

Table 6.14: User Profile management Functional Testcase 62

Table 6.15: User invitation management Functional Testcase 63

Table 6.16: User view list of users Functional Testcase 63

Table 6.17: Room creation Functional Testcase 63

Table 6.18: Search Rooms Functional Testcase 63

Table 6.19: Delete Rooms Functional Testcase 64

Table 6.20: Temporary Room creation Functional Testcase 64

Table 6.21: Private Room creation Functional Testcase 64

Table 6.22: Schedule Rooms Functional Testcase 64

Table 6.23: Video calling Functional Testcase 65

Table 6.24: Messaging audio/Text Functional Testcase 65

Table 6.25: Logout Functional Testcase 65

Table 6.26: Integration Testcase 65

Table 6.26: *Table 6.27: Automated testing* 66

# Introduction

Roomy is dynamic web application build for communication in the real time with themed chat rooms. This application helps us to connect with people having same interests. The key feature of roomy app is to create temporary chat rooms. Users can send audio messages and video calling features. The unique feature is temporary chat rooms. User can be able to discuss various topics in the rooms.

## Brief

The main goal of this project, a web application that enables real-time communication through themed chat rooms. The main goal of this platform is to provide is to connect the likeminded people.

## Relevance to Course Modules

Relevance to Course Modules in the Context of the Roomy Proposal.

### Database Management Systems:

Databases are one of the most important things for the roomy. Where the messages of the user will store it must be secure. Understanding database management systems is essential for *Roomy* as it involves storing and managing user information, chat histories, and room data. It is a cross-platform document- oriented NoSQL database.

### Web Development Technologies:

We Develop Roomy using MERN stack (Mongo DB, Express.js, React.js, and Node.js) for web development. And for mobile applications we use react native for frontend and for backend we use node.js we will learn about frontend development with React.js and for mobile we use react native backend development with Node.js and Express.js, and database integration with Mongo DB.

### Software Engineering Principles and Practices:

Roomy Chatrooms platform uses the software engineering principle in development cycle. The first phase is requirement gathering first we gather the requirement through different technique like interviewing and brainstorming. And after requirement gathering is completed. And we start development and testing and run the cycles.

### Human-Computer Interaction (HCI):

HCI principles play a most important role in designing a user-friendly interface for *Roomy*. Students will apply design principal design methodologies to create an intuitive design Roomy features like chat room discovery, profile management, and easy navigation.

1

### Project Management and Agile Methodologies:

Roomy uses Agile methods, focusing on small, regular updates and feedback. Students learn Agile skills like sprint planning, daily scrum, refining backlogs, and sprint reviews. This boosts their project management abilities.

### Security and Privacy in Software Development:

Security and privacy are the most important feature for Roomy, especially with user data and communication. Students will learn secure coding, authentication, and encryption to protect data. This helps prevent unauthorized users (access) and meets data protection rules

### Software Testing and Quality Assurance:

Testing is one of the important parts of *Roomy*’s development. Students will learn testing techniques.

## Project Background

The Roomy project is designed to provide users with a platform for real-time communication through themed chat rooms. Unlike traditional messaging platforms, Roomy focuses on temporary rooms created around specific interests, allowing users to engage in focused discussions. The idea behind the project is to foster dynamic and meaningful conversations based on shared topics, while offering both text and audio chat functionalities.

## Literature Review

Here is the literature review of the project Roomy.

Discord is a free communication app where millions of users connect and use messages, audios, and videos with each other. It’s a gaming platform. Discord can be used on every operating system like window, Linux, MacOS, iOS and used in systems like web, Android etc. Microsoft Teams a popular platform for online studying it provide abilities to users to schedule their room on basis of this users will share their assignments and provide sending messages with each other better providing screen sharing. Telegram is a popular platform for cloud-based messaging. It provides group chats, file sharing, and providing bots. its best for security purpose but it does not support temporary, and scheduling rooms like in roomy chatrooms. Google Chat is part of the Google platform where users can interact in groups and sent messages. It integrates seamlessly with other Google apps, making it a good choice for teams already using Google's ecosystem. However, it is primarily work-focused and lacks the casual, interest-based community-building features of Roomy. Roomy combines the best aspects of these platforms—structured communication from Slack, community focus from Discord, security and speed from Telegram, and scheduling flexibility absent in Microsoft Teams and Google Chat. Its unique focus on temporary chat rooms and interest-based discussions sets it apart as an innovative communication tool.

## Analysis from Literature Review

The analysis from the literature review provides valuable artifacts about the limitations of existing communication platforms like Slack and Discord, while identifying the opportunities for improvement on Roomy to address issues. Here's how the literature review analysis contributes to the Roomy project.

### Finding Weaknesses in Existing Platforms:

The literature reviews indicate there are several issues in these applications like slack and discord. Slack is great for the work environment but has no other social uses. Even though Discord is okay for communities, it's not great for quick, temporary chats. We found out there's a need for something different. That’s where Roomy comes Roomy is trying to be the app for quick, topic- based chat It's for people who want to jump in, talk about something specific, and then leave. to jump in, talk about something specific, and then leave. Roomy aims to be more active and engaging than the other apps.

### Comparison with Previous created System:

By comparing the limitations of existing platforms with the solutions proposed by Roomy, Slack organizes communication around work channels, while Discord is more suitable for long-term where social interaction can take place, Roomy allows users to create and join temporary rooms around specific interests. Roomy provides temporary rooms based on interests like Room scheduling & audio chat.

### Scope and Limitations Consideration:

Furthermore, the literature review considers current platforms their strengths and weaknesses including a particular focus on systems designed for long-term groups or workplace settings with types of conversations that tend to be longer in duration and/or time from one another, so it does not address spontaneous/just-in-time/temporary discussion support challenges. The findings from this analysis help influence the direction of Roomy, in that it is a network where the core need for on demand interest-based short-term interactions can be met but put our design energy toward its challenges such as temporarily moderating rooms and fostering user participation

## Methodology and Software Lifecycle for This Project

A To build Roomy properly, it's important to follow a clear plan and use a software development process that keeps the project on track. Meets stakeholder requirements and delivers a high-quality product. Here's a proposed methodology and software lifecycle for the Roomy project.

### Agile Software Development Methodology

Agile methodology is best for Roomy due to its force on iterative development, collaboration, and adaptability. Using agile methodology, the project remains

flexible and responsive to user feedback and changing requirements throughout its lifecycle

Characteristics of Agile Software Development Methodology

* + - * **Iterative Development:** Roomy will be created through iterative cycles. with each sprint delivering specific features like room creation, real-time chat, or audio chat functionalities. Each iteration will involve testing and review to ensure quality product is made.
      * **Continuous Feedback:** The process will continue a regular feedback loop, allowing stakeholders and users to share their ideas on newly developed features. This helps in confirmation of requirements and immediate modifications based on feedback. Make sure that roomy follow the customer need and expectation.
      * **Implementation:** Agile will be implemented using the Scrum framework. Each sprint will focus on delivering functional product, and the team will combine to adjust and improvements in each cycle. This iterative and collaborative approach ensures a robust and user-cantered development process for Roomy.

### Software Development Lifecycle (SDLC):

Here is the Software Development Lifecycle:

* + - * **Requirement Analysis:** requirements analysis is one of the most important features for roomy because project is totally dependent on requirements. This phase defines the core features of Roomy, such as room creation, real-time chat, audio chat, and moderation tools.
      * **Design:** design is the second most important part for roomy like system architecture database management, and user interface (UI) designs based on the requirements from the analysis phase. It has the predefine User interface design. For backend API integration etc.
      * **Development:** we develop a Roomy using mern stack. We made frontend using react.js backend with node.js and express.js and use mongo MongoDB for data storage. This phase also involves integrating real-time messaging, audio chat, and user profile customization features.
      * **Testing:** Testing to measure the quality of roomy. Thay contain unit testing for individual components, integration testing for the combination of module and user acceptance testing for following the user requirement. Check the scalability of the system the system will work on all devices.
      * **Deployment:** At this stage we will conduct the final testing before deployment. This stages how the system will work in real world environment.

# Problem Definition

Here is the problem definition of Roomy in which included Problem Statement and Deliverables.

## Problem Statement

The main problem in the current platform slack and discord. It does not provide temporary chat rooms.it focus on long term rooms. But our application provides temporary chat rooms that can be deleted automatically after a sometime. Our application provides a temporary, interest-based chat rooms.

### Lack of Temporary Chat Rooms:

Most platforms have on permanent existing chat rooms. It is very hard to make a temporary room for short-time discussions about specific topics if time expires room will delete and messages removed permanently.

### Limited Audio Chat for Short-Term Conversations:

There aren’t many options for users who want to join temporary voice chats for real-time discussions on specific topics.

### No Room Scheduling Options:

Most platforms don’t allow users to schedule rooms for specific times, while there is chance to miss the conversations in rooms.

### Challenges with Moderation:

Temporary chat rooms can be difficult to manage it. Slack and discord do not have strong tools for managing these kinds of chat rooms. Roomy will be a full- featured web app focused on temporary, topic-based chat rooms. It will offer real-time messaging and audio options, making it easy for user to interact simply and easily. Roomy will also include scheduling tools, strong changing features, and customization options to enhance the user experience.

## Deliverable and Development Requirements

The Deliverables for the Roomy.

### Fully Functional Web Application:

Create a web app that lets people set up, join, and manage chat rooms. The platform will support both short-term and long-term rooms, offering a simple way for users to stay connected

### User Registration and Login:

Implement a secure registration and login system in which user can login and use the system functionality. This will include security features like hashed, bcrypt.js which secure the password from hackers.

### Profile Management:

Register user can update the profiles like name address picture and bio information.

### Chat Room Creation and Management:

User can create a chat room based on specific topics. users set up chat rooms for specific time or topics. They can invite others, adjust room settings, and choose to make the room public or private.

### Temporary and Scheduled Chat Rooms:

Register user can create a temporary chat room and create a scheduling room.

### Real-Time Communication (Text, Audio, and Video):

Introduce audio chat rooms where users can engage in voice conversations in real time. Use features for joining, leaving, and muting audio like voice chat platforms. Users can use the video calling feature.

### Group Creation (Open and Private):

users create and create open or private groups. This setup lets people with similar interests connect and discuss in a secure platform.

### Search Chat Rooms:

Implement a search feature that allows users to quickly find chat rooms based on topics, interests, or specific keywords. It’s made a simplicity for user to easily find room by their names both experience and unexperienced users can excess easily.

### Delete Chat Rooms:

Provide the ability for room admin to delete chat rooms specific user have no ability and excess to delete room.

### Calendar and Scheduling Integration:

Integrate a scheduling feature that allows users to plan and coordinate discussions at specific times. This will help organize rooms for specific events, meetings, or topics.

## Current System

Many platforms now support real-time communication, each with its own features. Two of the most popular are Slack and Discord, which help teams collaborate and build communities using chat rooms or channels.

### Slack:

Slack is a business communication tool that helps team members stay connected and share important information. It allows people to work together more smoothly by creating a single space where everyone can collaborate and communicate easily.

Slack excels in professional settings, it lacks the casual, community-building aspect found in other platforms.

### Discord:

Discord is a free communication app where millions of users connect and used messages, audios, videos with each other it’s a gaming platform. Discord can be used on every operating system like window, Linux, MacOS, iOS etc and used in systems like web, Android etc

### Limitations of Current Systems:

Slack and Discord are good for communication, but they have some limits that Roomy wants to fix. Slack’s free version has restrictions that make it hard for small groups. It’s better for long-term projects than for quick chat rooms. Discord, on the other hand, isn’t great for professional settings. It focuses on voice chats and is made for gamers. Plus, neither platform makes it easy to set up and schedule temporary chat rooms for specific talks or events, which is important for some users.

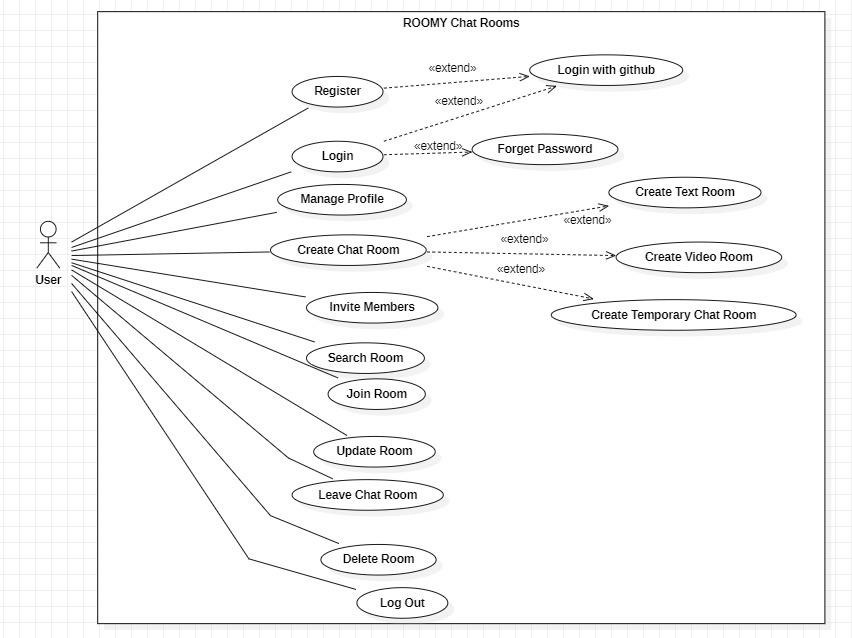
### Roomy's Improvement:

Roomy is made to fix these issues. It lets users create temporary chat rooms. They can also schedule talks using a built-in calendar. Users can chat with text, audio, or video easily. Roomy is good for both work and casual chats. It’s a better option for people who want quick discussions based on interests. Plus, it gives more control over how the rooms are set up.

# Requirement Analysis

Here is a Use case Diagram of Roomy.

## Use Cases Diagram(s)

****

*Fig 3.1 Use Case Diagram*

## Detailed Use Case

Here is the Detailed Use Case Diagram of Roomy.

### Use Case Description for Register:

*Table 3.1 Register*

|  |  |
| --- | --- |
| **Use case ID** | UC -01 |
| **Use case Name** | Register |
| **Actors** | User |
| **Description** | User opens Roomy homepage and want to create its account, click at the sign-up button, and creates its account to use Roomy app. |
| **Trigger** | The user wants to create a roomy account. |
| **Preconditions** | The user access to internet.  The user has an email address which is not used previously |
| **Postconditions** | The user now creates a new Roomy account. The user has access of his account and logged in. |
| **Normal Flow** | The user opens Roomy web application homepage. The user then enters the email address and password. The email and password are then validated.  The password meets the required criteria. The user is given access to its account. |
| **Alternative flow Alternative flow 1 (invalid**  **email)** | The user given invalid email.  The user is asked to provide a valid email.  The provide a valid email and the whole process repeats. |
| **Alternative flow Alternative flow 2 (username already taken)** | The user enters a username.  The username is already taken by someone else. Please enter another user name. |
| **Exceptions** | System overload can occur at any time.  If it occurs the system sends user an error message and asks user to try again after a while. |
| **Business Rules** | The Password has at least (one capital letter, one small letter, one number and one special letter).  Email addresses should be unique. |
| **assumptions** | The user has a computer system that can run the roomy webApp.  The user has the internet access.  The user knows how to use internet navigations. |

### Use Case Description for Sign in:

*Table 3.2 Sign In*

|  |  |
| --- | --- |
| **Use case ID** | UC -02 |
| **Use case Name** | Sign In |
| **Actors** | User |
| **Description** | User opens Roomy’s homepage and want to login to its Roomy’s account.  User clicks on Sign-In button.  The system granted access to its Roomy account.  If the user wants to log in to Roomy account through its GitHub, it just has to click on log in with GitHub button. The user is given access to its account. |
| **Trigger** | The user wants to access its Roomy account. |
| **Preconditions** | The user has a valid email address. |
| **Postconditions** | The user |
| **Normal Flow** | The user opens Roomy app homepage. The User enters the email and password. The user clicks on the login button.  User is verified successfully.  User is directed to its dashboard on Roomy app. |
| **Alternative flow Alternative flow 1 (invalid email)** | The email provided by the user is incorrect. The user is asked to provide the correct email.  The user provides the correct email and the whole process repeats. |
| **Exceptions** | In case of system overload please try again latter. |
| **Business Rules** | For sign in there should be only three attempts.  If the user is in active for 5 minutes the user account will be logged out. |
| **Assumptions** | The user has internet access.  The user remembers their registered email and password. |

### Use Case Description for Manage Profile:

*Table 3.3 Manage Profile*

|  |  |
| --- | --- |
| **Use case ID** | UC -03 |
| **Use case Name** | Manage Profile |
| **Actors** | User |
| **Description** | The user wants to update their profile for the Roomy account. |
| **Trigger** | The user wants to manage its profile on Roomy account or update its account information. |
| **Preconditions** | The user has a valid Roomy account. |
| **Postconditions** | The user modifies its profile based on its likings and interest. |
| **Normal Flow** | The user has already logged into its Roomy account and has access to its profile settings.  The user chooses the area to modify (e.g., edit bio, change profile picture, update interests).  The user edits its profile information. This can be editing text, uploading images, or changing a list.  The user saves the changes by clicking a “Save Changes” button.  The system updates its profile information according to user wishes. |
| **Alternative flow Alternative flow 1 (invalid**  **Information)** | The user changes are invalid (e.g., exceeding character limits, inappropriate content),  An error message is displayed. The user corrects the changes. It again submits the changes. |
| **Alternative flow Alternative flow 2 (Image**  **Upload Error)** | There comes error during uploading image (e.g., file format not supported, exceeding file size limit),  error message is displayed to user.  The user chooses a valid image and submit again. |
| **Exceptions** | In case of system overload please try again latter. |
| **Business Rules** | All the text fields should have character limits.  The system should enforce limitations on image file size and format (e.g., JPEG, PNG). |
| **Assumptions** | The user knows the basic internet use.  The user has the necessary resources for profile updates (e.g., image files for profile picture). |

### Use Case Description for Create Chat room:

*Table 3.4 Create Chat Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -04 |
| **Use case Name** | Create Chat Room |
| **Actors** | User |
| **Description** | The user wants to create an account on Roomy app. The room can be text, audio or video type. |
| **Trigger** | The user wants to create a space for focused conversations with other users in the world who has the same interest.  The users can choose the type of space for it. |
| **Preconditions** | The user has internet access.  The user has a valid email address. |
| **Postconditions** | A chat room is created on the Roomy app. The user is given authorities of room creator.  According to the chosen settings the room may be text- based, audio calls or video. |
| **Normal Flow** | The user opens Roomy app dashboard. Clicks on the “Create Chat Room” button.  The user gives details of the room especially the name of the room.  The user configures the room settings (text, audio, or video). This determines the facilities available within the room.  The user can choose the nature of room i.e. public or private. If the room is private the room creator set a password for room security.  The room creator confirms the details and settings by clicking on “Create Room”.  The system creates the room according to the details given by the room creator.  The user is automatically joined in the room. |
| **Alternative flow Alternative flow 1 (Invalid Room Name)** | The user enters room name invalid (e.g. exceeding character limit etc).  An error message is displayed to user to enter a valid name. The user enters the correct name and the whole process repeats. |
| **Alternative flow** | The name provided by the user is already taken. The user re- enters the name and resubmit it. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | Room name should follow specific guidelines like name length etc.  Inappropriate language must be restricted in room names. There should be a limited number of rooms a user can create. |
| **Assumptions** | The user knows the purpose of room he is creating.  The user understands the topic and theme of the room and communication format (text, audio, or video). |

### Use Case Description for Invite Members

*Table 3.5 Invite Members*

|  |  |
| --- | --- |
| **Use case ID** | UC -05 |
| **Use case Name** | Invite Members |
| **Actors** | Room creator |
| **Description** | The user wants to invite other users to join his created room. |
| **Trigger** | The room creator desires to invite users to participate in the conversation. |
| **Preconditions** | The user is a room creator.  The user has full authorities of the room. The room should be private. |
| **Postconditions** | The users who are invited get a notification or message. The invited members can accept or decline the invite. |
| **Normal Flow** | The user accesses its created chat room.  The user goes to the “Invite Members” section.  The room creator copies the link and send it to the desired contact through WhatsApp or email.  The invited member of Roomy clicks on the link and is joined into the specific room. |
| **Alternative flow Alternative flow 1 (User not found)** | The room creator wants to send invite to other, but the other user doesn’t exist.  The system shows an error message to the room creator indicating that the user doesn’t exist. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | Members of the private room does not have permission to send invites to other members.  There should have limitations on number of invites to prevent spam. |
| **Assumptions** | The user knows the username or email of the specific member he/she wants to invite.  The room creator knows all the protocols of adding members through invitations. |

### Use Case Description for Search Rooms:

*Table 3.6 Search Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -06 |
| **Use case Name** | Search Room |
| **Actors** | User |
| **Description** | The user wants to explore chat rooms of its interests. Users can find rooms of its specific interests and join conversations of its likings. |
| **Trigger** | The member of Roomy chat rooms wants to explore a chat room of its interest. |
| **Preconditions** | The use has access to the internet.  The user is a member of Roomy chat rooms. The user is logged in to the app. |
| **Postconditions** | The user finds the chat room of its interest. The user joins the chat room.  The process is facilitated by the system. |
| **Normal Flow** | The user logs into its Roomy account.  The user search for specific chat room of its interest using some specific keywords.  The system fetches the specific chat room from data base. The user selects its desired chat room.  The user is provided some details about that room.  The user joins the chat room by clicking on “Join Room” button.  Finds the “Join Room” option. |
| **Alternative flow Alternative flow 1 (Confirmation Cancellation)** | The user enters search query and click on search button. The system processes the search request but finds no chat rooms matching the criteria.  The system shows a message indicating no matching chat rooms were found and suggests alternative search terms or criteria. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | The system might limit the number of search results displayed at once to ensure performance.  Search functionality should account for variations in spelling and synonyms to improve search accuracy. |
| **Assumptions** | The user understands how to use the search functionality within the Roomy platform.  The user has a clear idea of the topics or themes they are interested in finding chat rooms for. |

### Use Case Description for join Rooms:

*Table 3.7 Join Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -07 |
| **Use case Name** | Join Room |
| **Actors** | User |
| **Description** | This use case describes the process of user to join a chat room of its interest on Roomy app, that allows the user to participate in that specific room. |
| **Trigger** | The user finds a room and want to join it. |
| **Preconditions** | The use has access to the internet.  The user is a member of Roomy chat rooms. The user is logged in to the app. |
| **Postconditions** | The user finds the chat room of its interest. The user joins the chat room.  The process is facilitated by the system. |
| **Normal Flow** | The user goes to the “Join Room” section and clicks on it. The system evaluates the nature of the room either it is a private or public room.  If the room is public the user is given access to the room and private they asked for password.  The user enters password.  The system evaluates the password.  If it enters the right password, the system allows it to enter into the room.  There is also another option for user to contact the room creator for the password.  The room creator gives the user password, and the user joins the room. |
| **Alternative flow Alternative flow 1 (Invalid Password)** | The user enters a wrong password.  The user is asked to enter a valid password.  Also, the user is asked to contact the room creator. The user contacts the room creator.  The room creator gives the user password, and the user joins the room. |
| **Exceptions (Exception 1) (System**  **Overload)** | In case of system overload, please try again latter. |
| **Business Rules** | There should be a limited number of members in a room to enhance the performance of the room.  The password should meet specific criteria. |
| **Assumptions** | The user knows the process of joining a chat room. The user has the password to join private chat rooms. |

### Use Case Description for Schedule Chat Rooms:

*Table 3.8 Schedule Chat Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -08 |
| **Use case Name** | Schedule Chat Room |
| **Actors** | User |
| **Description** | The user wants to schedule a room in future.  It is done so that all the participants of the room must plan for the room and make the necessary arrangements. |
| **Trigger** | Thes user want to create a room for the near future for some predefined purpose, also informing other members to make the necessary arrangements and plan for the chat  room meeting. |
| **Preconditions** | The use has access to the internet.  The user is a member of Roomy chat rooms. The user is logged in to the app. |
| **Postconditions** | The user creates Chat room for future use. The chat room is scheduled.  The chat room can be of text, audio or video conferencing. |
| **Normal Flow** | The user goes to the create Chat Room section of the Roomy app page.  The user clicks on the Schedule Chat Room option.  The user provides all the necessary information about that room to be created.  The user selects the date and time for the room. The user confirms the details of the chat room  The user then notifies other users about the scheduled room.  The system creates the scheduled chat room successfully. |
| **Alternative flow Alternative flow 1 (Schedule Conflict)** | The selected date and time is not in selected range (e.g. exceeding maximum scheduling range).  The system display an error message. |
| **Alternative flow Alternative flow 2 (Room Name Already**  **Taken)** | The name chosen by the user is taken by someone else. The system shows an error message to the user indicating that the name is already taken.  The user enters another name, and the room is scheduled accordingly. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | There user can create a limited number of chat rooms. Room names must follow a specific naming convention for length and characters. |
| **Assumptions** | The user understands the working of scheduled chat rooms. |

|  |  |
| --- | --- |
|  | The user has a clear idea of topic of its scheduled chat  room, and the mode of chat room, and the intended duration for the conversation. |

### Use Case Description for update chat room:

*Table 3.9 update chat Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -09 |
| **Use case Name** | Update Chat Room |
| **Actors** | Room creator |
| **Description** | This use case shows the process of updating an existing chat room by the room creator. |
| **Trigger** | The room creator wants to make some changes to the room the user created. |
| **Preconditions** | The user already created the chat room it wants to update |
| **Postconditions** | The room creator successfully updates all the desired sections of the room it previously created. |
| **Normal Flow** | The room creator navigates to the room it previously created.  The room creator clicks on the update room section.  The system redirects the room creator to the settings of the room.  The room creator updates the settings of the room as it desires.  Clicks on the update changes.  The system updates the room successfully. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | The updates user wants to do should not exceeds the guidelines mentioned above. |
| **Assumptions** | The user understands all the setting policy. The user knows how to navigate in setting. |

### Use Case Description for Leave Room

*Table 3.10 Leave Room*

|  |  |
| --- | --- |
| **Use case ID** | UC -10 |
| **Use case Name** | Leave Room |
| **Actors** | User |
| **Description** | This use case describes the process of a user want to leave a chat room it is a member of. |
| **Trigger** | The user decides to leave a chat room it is a member of. |
| **Preconditions** | The user has internet access.  The user has a valid Roomy account.  The user is successfully logged into the application.  The user is a member of the chat room it wishes to leave. |

|  |  |
| --- | --- |
| **Postconditions** | The user leaves the chat room successfully.  The user has no longer access to the room's conversations. |
| **Normal Flow** | The user goes to leave chat room section it. The user clicks on the "Leave Room" button.  The system processes the request and removes the user from the chat room.  The user is redirected to its account dashboard. |
| **Alternative flow (Confirmation Cancellation)** | The user clicks on “leave chat room”.  The system ask to the user to confirm the leaving action. The user chooses to cancel the confirmation.  The leave room process is cancelled.  The user remains a member of that chat room |
| **Exceptions (System**  **Overload)** | In case of system overload, please try again latter. |
| **Business Rules** | Users can be asked to give a reason for leaving the room, depending on the room's settings.  The user should be informed that it will leave all the privileges permanently. |
| **Assumptions** | The user understands the operations of leaving a chat room. The user has navigated the Roomy platform to locate the "Leave Room" option. |

### Use Case Description for Delete chat Rooms

*Table 3.11 Delete chat rooms*

|  |  |
| --- | --- |
| **Use case ID** | UC -11 |
| **Use case Name** | Delete chat rooms |
| **Actors** | Room creator |
| **Description** | This use case shows the process for a room creator to delete a room from the Roomy platform. |
| **Trigger** | The room creator wants to delete the room as there is no longer need for the room. |
| **Preconditions** | The user has a valid Roomy account.  The user is successfully logged into the Roomy app. The user is the creator of the room. |
| **Postconditions** | The chosen room is permanently deleted from the Roomy app.  All data, including chat history, member lists are removed. All other members can’t access that specific room. |
| **Normal Flow** | The room creator accesses the room settings. It chooses a room to be permanently deleted.  The system prompts user to confirm the room deletion. The room creator confirms room deletion.  The system deletes all room data.  The system prompts user for the successful deletion of the room. |

|  |  |
| --- | --- |
| **Alternative flow**  **(Confirmation Cancellation)** | The system prompts room creator to confirm the deletion or cancel it.  The room creator cancels to process.  The system aborts the cancellation process. |
| **Exceptions** | In case of system overload, please try again latter.  If the user attempting to delete the room is not the owner, the system displays an error message showing an invalid action. |
| **Business Rules** | Deleting room is a permanent action.  Room creator should confirm before deleting a room, as data recovery cannot be possible. |
| **Assumptions** | The room creator has considered the decision to delete. |

### Use Case Description for Logout

*Table 3.12 log Out*

|  |  |
| --- | --- |
| **Use case ID** | UC -12 |
| **Use case Name** | Log Out |
| **Actors** | User |
| **Description** | This use case describes the process of a user logging out of the system. |
| **Trigger** | The user wants to end the section of use of the Roomy app. |
| **Preconditions** | The user is no longer connected to the Roomy app. Any information of the user's account is no longer accessible on the device. |
| **Postconditions** | The user has no longer access to its account features. |
| **Normal Flow** | The user goes to profile settings within the Roomy app. The user identifies "Log Out" button.  The user selects the "Log Out" option. The system asks the user for confirmation. The user confirms the logout.  The system successfully ends the user's session. |
| **Alternative flow (Confirmation Cancellation)** | The user chooses to cancel the logout process\ The system cancels the logout action  The user remains logged in. |
| **Exceptions** | In case of system overload, please try again latter. |
| **Business Rules** | The user might be logged out of its account if it is not active for a while.  The user once logged out cannot access application features until it again log in. |
| **Assumptions** | The user knows the purpose of logging out and its impact on its ongoing session. |

## Functional Requirements

The functional requirements for Roomy outline the key features and capabilities the platform needs to provide for seamless communication. These requirements ensure that users can manage their accounts, create and join chat rooms, and engage in secure text, audio, and video communication. The following are the functional requirements for Roomy.

### User Registration and Authentication:

Provide a secure mechanism for new users to sign up and create an account on Roomy.

Allow existing users to log in securely, enabling them to access chat rooms and participate in conversations.

### Profile Management:

Enable users to manage and personalize their profiles by adding information such as bios, interests, and profile pictures.

### Chat Room Creation:

Allow users to create text chat rooms based on various topics of interest. Enable registered users to create dedicated audio and video chat rooms for more immersive communication.

### Temporary Chat Room Creation:

Provide users the ability to create temporary chat rooms for short-term discussions that can be scheduled to expire after a specific time.

### Invitation Management:

Allow users to invite others to join chat rooms, enabling collaborative and diverse discussions.

### Search and Join Chat Rooms:

Provide a search feature for users to find chat rooms by name or topic of interest. Enable users to join existing chat rooms to participate in ongoing discussions.

### Room Scheduling and Deletion:

Allow users to schedule chat rooms for future events or discussions, ensuring participants are notified and prepared. Provide group creators with the ability to delete chat rooms once the conversation is over or the room is no longer relevant.

### Real Time Communication

Provide real-time text, audio, and video communication within chat rooms, ensuring smooth, multi-modal conversations

### Log Out and Account Security:

Ensure users can securely log out from their accounts to maintain their privacy and protect their information

## Non-Functional Requirements

Nonfunctional requirements describe how Roomy should perform and the quality it must maintain. These focus on things like speed, security, ease of use, and scalability. Below are the non-functional requirements for Roomy.

### Performance

* + - * **Response Time:** Roomy should load quickly when users navigate between chat rooms, send messages, or start video/audio calls.
      * **Scalability:** The system should be able to handle more users and chat rooms as the platform grows without slowing down.
      * **Capacity:** Roomy should support many users at the same time, especially during busy periods, to ensure smooth communication**.**

### Security

* + - * **Data Protection**: Sensitive information like passwords, messages, and video/audio chats should be encrypted to keep it secure.
      * **Access Control**: Only authorized users should be able to access certain features and rooms based on their roles.
      * **Activity Logging**: The system should keep track of important actions to detect any suspicious activities or security breaches.

### Usability

* + - * **Easy to Use**: Roomy’s design should be simple and easy for users to understand and navigate.
      * **Accessibility**: The platform should be accessible to people with disabilities, following basic accessibility guidelines.
      * **Help and Support**: Provide user guides, tutorials, and troubleshooting help so users can easily understand how to use Roomy.

### Reliability

* + - * **Error Handling**: Roomy should be able to handle small errors without crashing, so users can keep chatting without disruptions.
      * **High Availability**: The system should be up and running most of the time, with only minimal downtime for updates or maintenance.
      * **Data Safety**: All user information, including messages and profile details, should be protected from loss or corruption.

### Compatibility

* + - * **Works on All Browsers**: Roomy should work smoothly across different web browsers like Chrome, Firefox, Safari, and Edge.
      * **Device-Friendly**: Roomy should work well on various devices, including computers, tablets, and smartphones, with consistent experience on each.

### Scalability

* + - * **Add More Resources**: Roomy should be able to grow by adding more computing power if needed, to handle increased user demand.
      * **Expand the System**: It should be easy to add more servers to manage a larger user base and keep Roomy running smoothly as it grows.

### Interoperability

* + - * **Integration:** Roomy should work well with other systems and services, like calendars, external communication platforms, or APIs, for added features.

### Maintainability

* + - * **Easy to Update:** The system’s code should be written in a way that makes it easy to update, fix bugs, and improve features.
      * **Clear Documentation:** There should be detailed documentation to help developers maintain and update Roomy easily in the future**.**

# Design and Architecture

Here is detailed Design and Architecture of Roomy.

## System Architecture

Here is the System Architecture of Romy.

### Frontend (React.js)

* + - * **Description:** This module represents the user interface of the Roomy chat application. It is developed using React.js, a powerful JavaScript library known for building dynamic and interactive web applications. The frontend module provides the platform for users to access all features of Roomy through a modern and user- friendly interface.
      * **Functionality:** The front end handles the user interaction with Roomy's key features, such as registering new users, logging in, creating or joining chat rooms, and engaging in text, audio, and video communications. It allows users to manage their profiles, search for chat rooms, schedule future chats, and receive notifications. Designed to be responsive, the interface adapts to various screen sizes, offering a seamless experience across desktop and mobile devices.

### Backend (Node.js + Express.js)

* + - * **Description:** This module constitutes the server-side logic of the Roomy application. Built using Node.js and Express.js, it handles the data flow between the frontend and the database, manages requests, and executes business logic. It ensures secure communication and data processing behind the scenes.
      * **Functionality:** The backend is responsible for handling CRUD operations for user accounts, chat rooms, and messaging data. It processes requests from the frontend, ensuring that actions like creating rooms, sending messages, or updating profiles are completed in real-time. Additionally, the backend orchestrates secure user authentication and data retrieval, integrating with MongoDB to store and retrieve all user and chat room information.

### Authentication Service

* + - * **Description:** This module handles the authentication of the system. It ensures that only authorized users can log in and access the system's features.
      * **Functionality:** The authentication service manages user login and registration, verifies credentials using token-based authentication (such as JWT), and enforces secure access to the platform. It allows users to log in, log out, and securely manage their accounts, ensuring that sensitive user data is encrypted, and access is role-based to maintain system integrity.

### Chat Room Service

* + - * **Description:** This module is responsible for managing chat rooms within Roomy. It handles the creation, modification, and deletion of chat rooms, both temporary and permanent.
      * **Functionality**: The chat room service allows users to create chat rooms for text, audio, and video communication. Users can also schedule future chat rooms and invite others to join. This service ensures users can manage chat room settings, search for specific rooms, update room policies, and delete rooms when they are no longer needed. It provides a dynamic environment for real-time collaboration and conversation.

### Communication Service

* + - * **Description:** This module facilitates real-time communication between users in Roomy. It supports text messaging, audio, and video communication across various chat rooms.
      * **Functionality:** The communication service ensures that messages are delivered instantly between users, whether they are sending text, audio, or video messages. It integrates WebSocket technology for real-time, two-way communication, providing fast and efficient interaction. Notifications and reminders for scheduled rooms are also managed by this service, keeping users informed and engaged.

### User Profile Service

* + - * **Description:** This module manages user profiles, allowing users to customize and update their personal information on the Roomy platform.
      * **Functionality:** The user profile service enables users to create and edit profiles by adding details such as names, bios, and profile pictures. Users can update their preferences, manage privacy settings, and view their activity history. This service ensures that personal information is securely stored and accessible for modification whenever necessary.

.

### Group Management Service

* + - * **Description:** This module manages the creation and maintenance of user groups, including open and private groups within Roomy.
      * **Functionality:** The group management service allows users to create both public and private groups for discussions. Group creators can manage membership, set privacy levels, and invite others to join. This service ensures that users can leave or join groups, update group settings, and maintain the exclusivity of private groups by controlling access.

### Database (MongoDB)

* + - * **Description:** This module handles the storage of all data within the Roomy system. Built using MongoDB, it provides a NoSQL database that stores user profiles, chat rooms, messages, and other relevant information.
      * **Functionality:** The database stores and retrieves all system data, including chat logs, user details, and room information. The MongoDB database is integrated with the backend to provide fast access to real-time data across all aspects of the platform.

### 4.1.9 Architecture Diagram

*Fig 4.1: Architecture diagram*

## Data Representation

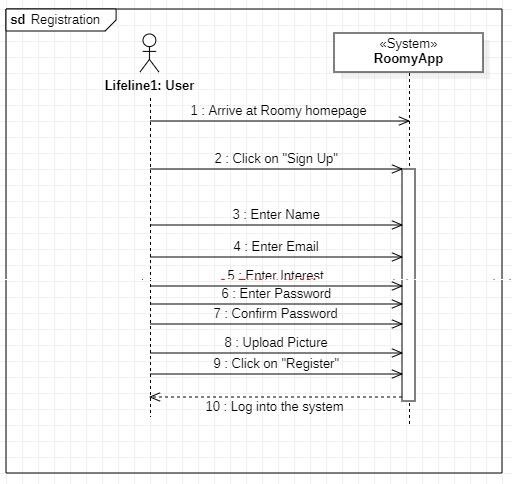
Here is detailed Design and Architecture of Roomy.

### System Sequence Diagram

*Fig 4.2: System Sequence diagram*

### Sequence Diagram for Registration

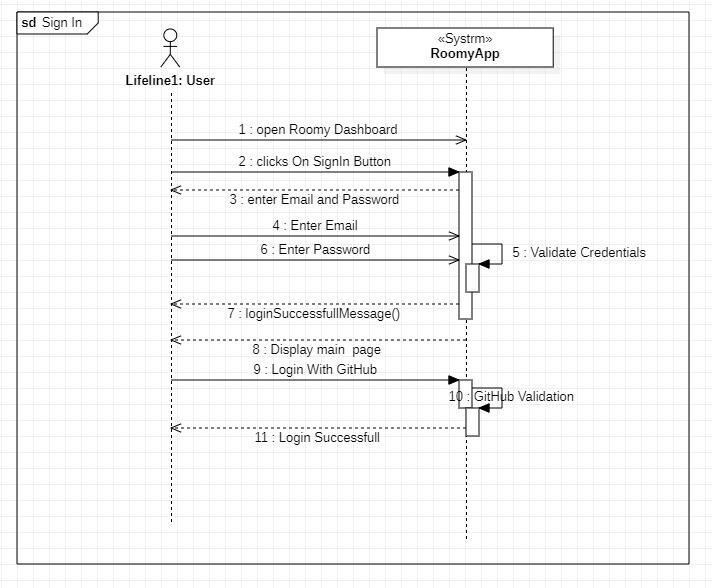
This SSD illustrates the process by which a new user signs up for Roomy by providing necessary information such as an email address and password, which the system then validates before creating a new user account and sending a confirmation response



*Fig 4.3: Sequence diagram for Registration*

### Sequence Diagram for Sign in

This SSD outlines how an existing user logs into their Roomy account by entering their email address and password, with the system validating the credentials and granting access upon successful validation

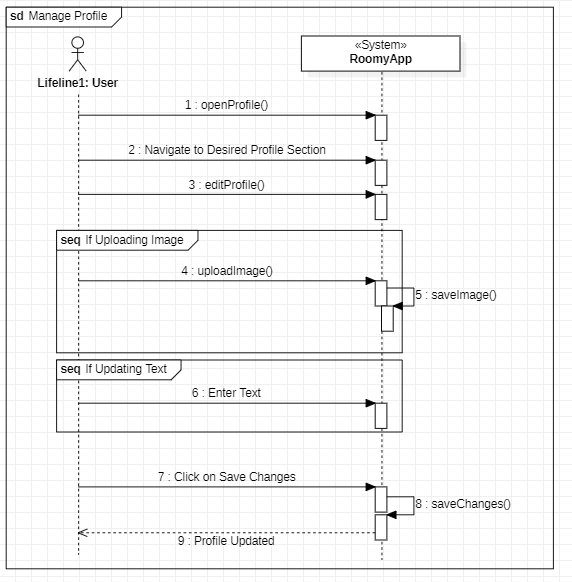


*Fig 4.4: Sequence diagram for Sign in*

.

### Sequence Diagram for Manage Profile

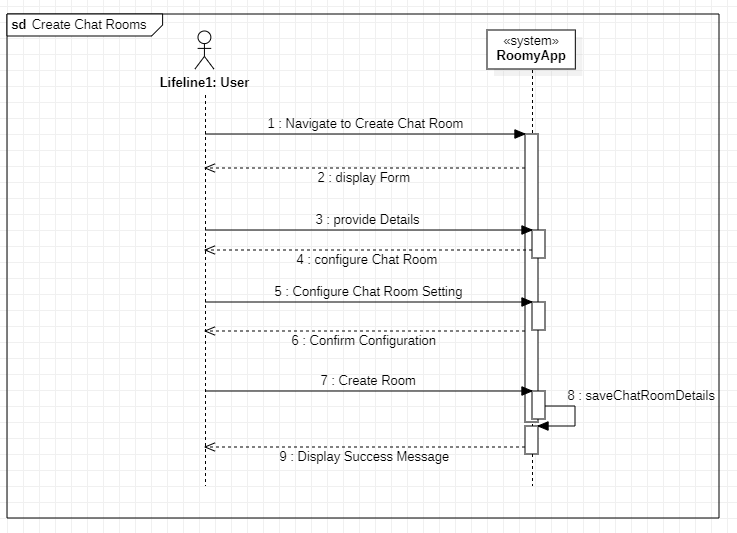
This SSD details the steps a user takes to customize their profile by updating their interests, bio, and profile picture, which the system then validates and saves before confirming the update to the user.



*Fig 4.5: Sequence diagram for Manage Profile*

### Sequence Diagram for Create Chat Rooms

This SSD shows how a user creates a new chat room by providing a room name, which the system then uses to generate the chat room and notify the user of its successful creation

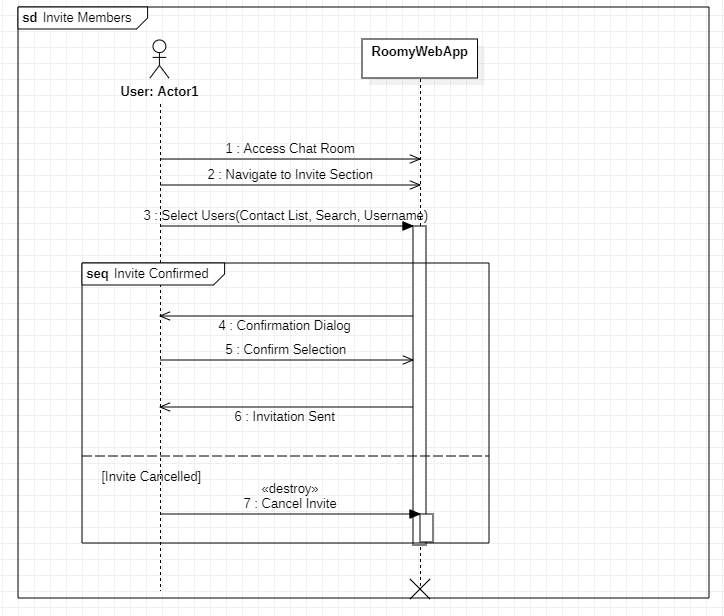


*Fig 4.6: Sequence diagram for create Chat Rooms*

.

### Sequence Diagram Invite members

This SSD depicts how a user invites other members to a chat room by generating an invitation link or sending an email invite, which the system processes and sends to the invited users

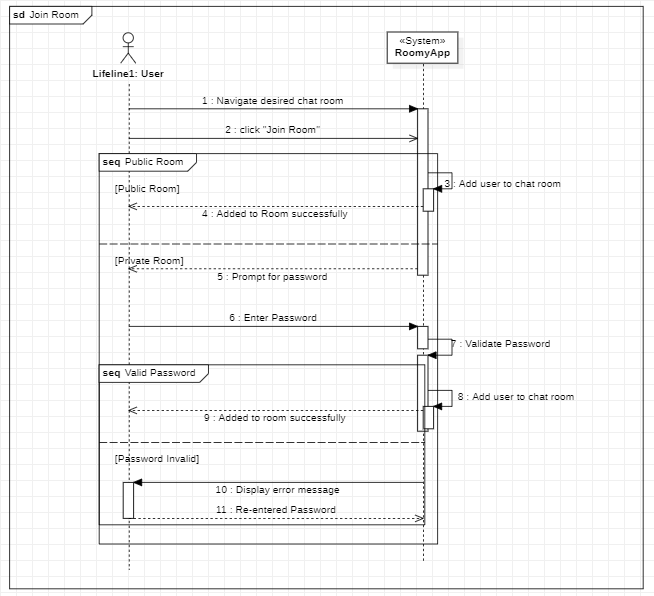


*Fig 4.7: Sequence diagram for invite members*

.

### Sequence Diagram for join chat Room

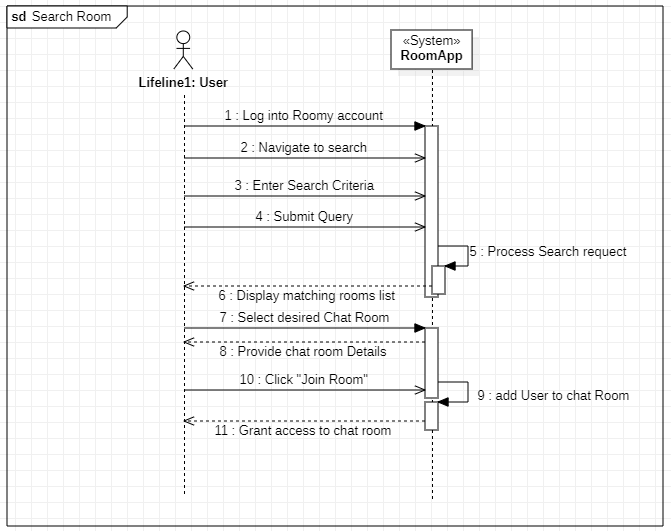
This SSD illustrates how a user joins a chat room via an invitation link, with the system validating the link and granting access to the chat room.



*Fig 4.8: Sequence diagram for join chat room*

### Sequence Diagram for Search Rooms

This SSD outlines the process a user follows to search for chat rooms by name, with the system retrieving and displaying relevant chat rooms based on the search query.



*Fig 4.9: Sequence diagram for search Rooms*

### A diagram of a system Description automatically generatedSequence Diagram for update Room

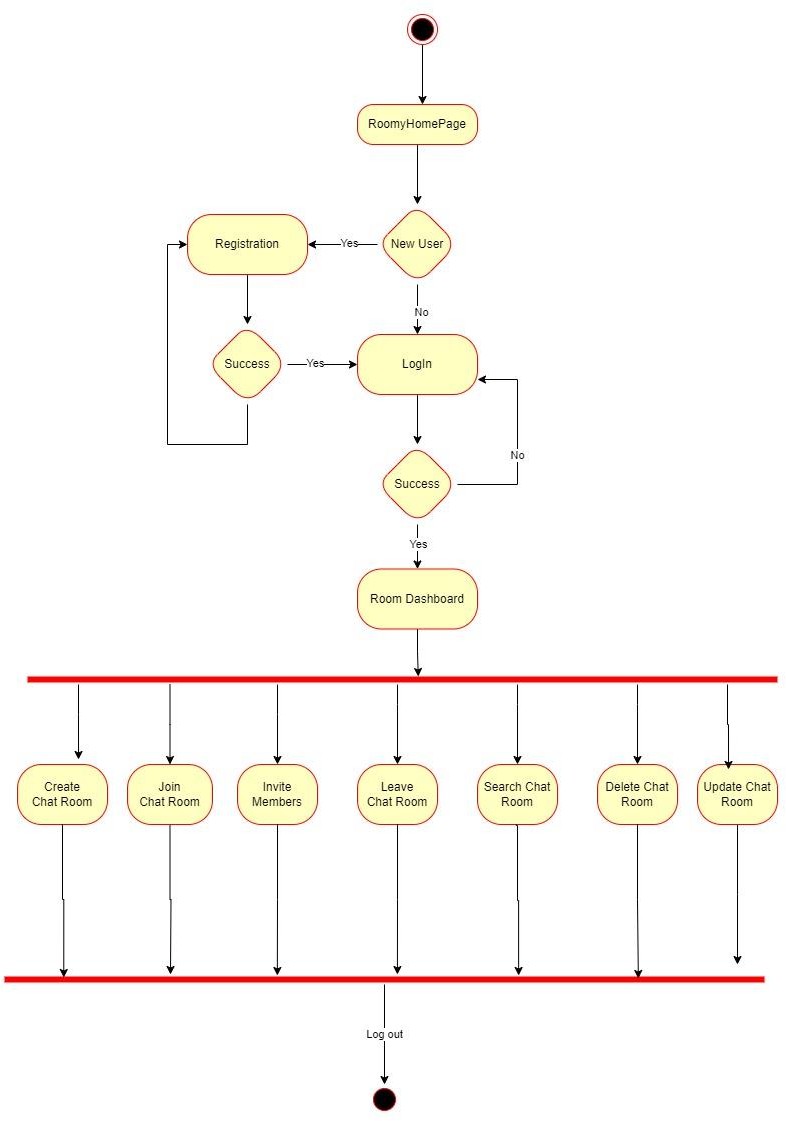
*Fig 4.10: Sequence diagram for update room*

### A screenshot of a chat room Description automatically generatedSequence Diagram for Delete Chat Room

*Fig 4.11: Sequence diagram for delete chat room*

## Process Flow/Representation

Here is a process flow diagram for Roomy.

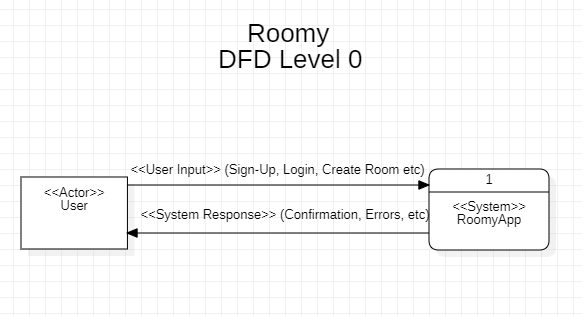


*Fig 4.12 process flow diagram*

## Design Models [along with descriptions]

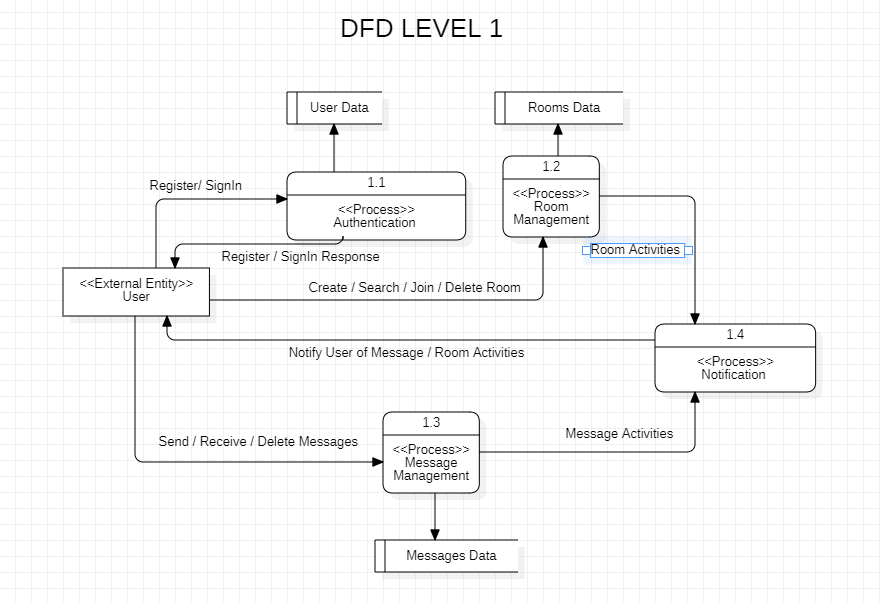
Here is Design Models with different levels of diagram

### Data Flow Diagram Level 0:

****

*Fig 4.13: Data Flow Diagram level 0*

### Data Flow Diagram Level 1:

****

*Fig 4.14: Data Flow Diagram level 1*

### Data Flow Diagram Level 2:

*Fig 4.15: Data Flow Diagram level 2*

# Implementation

This chapter will discuss implementation details.

## Algorithm

### Pseudo Code for Register User:

// Frontend (React.js)

// Function triggered when user submits the registration form function registerUser(userData) {

// Prepare data for submission formData.append('email', userData.email); formData.append('password', userData.password); formData.append('interest', userData.interest.join(',')); if (userData.image) { formData.append('profilePicture', userData.image);

}

// Send data to backend fetch(['http://localhost:5000/api/auth/createUser',](http://localhost:5000/api/auth/createUser%27) { method: 'POST',

body: formData,

})

.then((response) => response.json())

.then((data) => {

// Update UI based on response if (data.success) {

saveUserData(data.token, data.user); navigateTo('/create\_room'); displaySuccessMessage('Registration successful!');

} else {

displayErrorMessage(data.message || 'Registration failed.');

}

})

.catch(() => {

displayErrorMessage('An error occurred while processing your request.');

});

}

// Backend (Node.js with Express.js)

// Endpoint for registering a user app.post('/api/auth/createUser', (req, res) => {

const { name, email, password, interest } = req.body;

// Validate data

if (!name || !email || !password) {

return res.json({ success: false, message: 'Missing required fields.' });

}

### Pseudo Code For Login:

// Frontend (React.js)

// Function triggered when user submits login form function loginUser(credentials) {

// Send data to backend fetch('/api/auth/login', { method: 'POST', headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(credentials),

})

.then(response => response.json())

.then(data => {

// Update UI based on response if (data.token && data.user) {

storeTokenAndUserData(data.token, data.user); navigateToDashboard(); showLoginSuccessMessage();

} else {

displayLoginError(data.error || 'Login failed.');

}

})

.catch(error => {

displayLoginError('An unexpected error occurred during login.');

});

}

### Pseudo Code For Register User:

// Frontend (React.js)

// Function triggered when user submits profile update function submitProfileUpdate(userData) {

// Prepare data to send to backend

Create FormData object and append user details (name, email, password, bio, interest).

If a new profile picture exists, append it to FormData.

// Send data to backend fetch('/api/auth/updateuser', { method: 'PATCH',

headers: {

'Authorization': `Bearer ${token}`,

},

body: FormData,

})

.then(response => response.json())

.then(data => {

// Update UI based on response if (data.user) {

Display success message to the user.

Update profile picture and other details in the UI. Save updated user data to local storage.

} else {

Display error message with details from the response.

}

})

.catch(error => {

Display generic error message to the user.

});

}

// Validate incoming data

Validate fields (name, email, password, bio, interest). Handle file upload if a profile picture is included.

// Update profile in the database if (valid data) {

Update user data in the database using userId from the token. Return success response with updated user data.

} else {

Return failure response with validation error message.

}

});

## External APIs

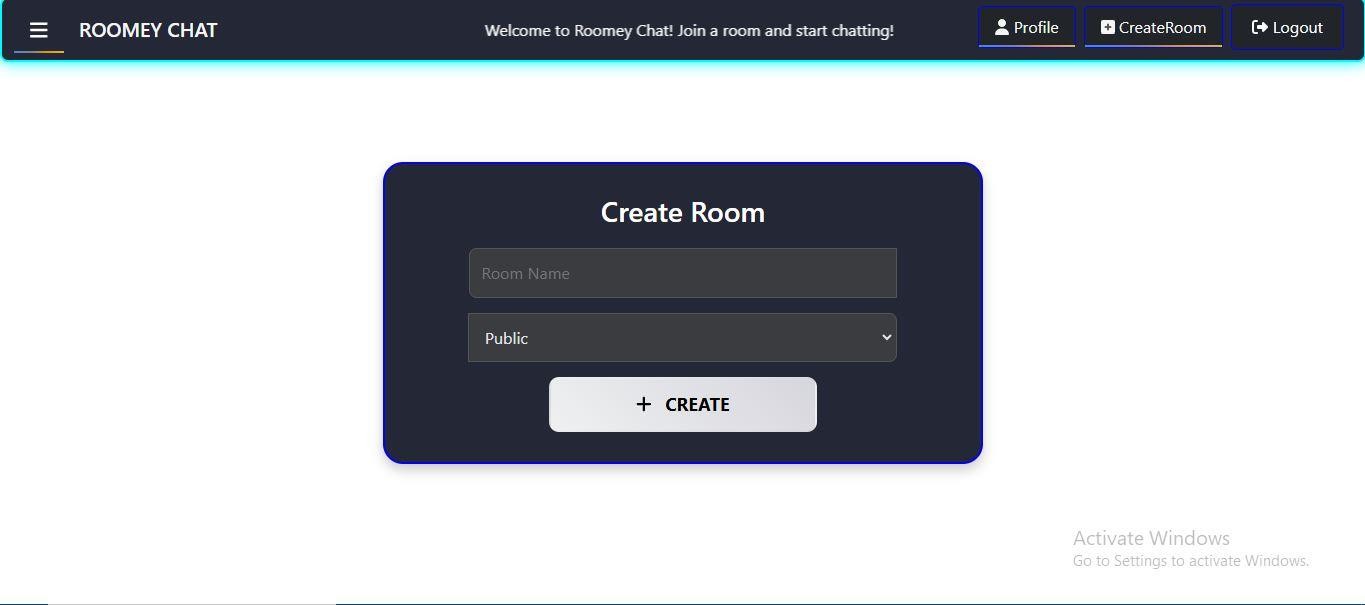
Table 5.1 shows that the Api list.

*Table 5.1: Details of APIs used in the project*

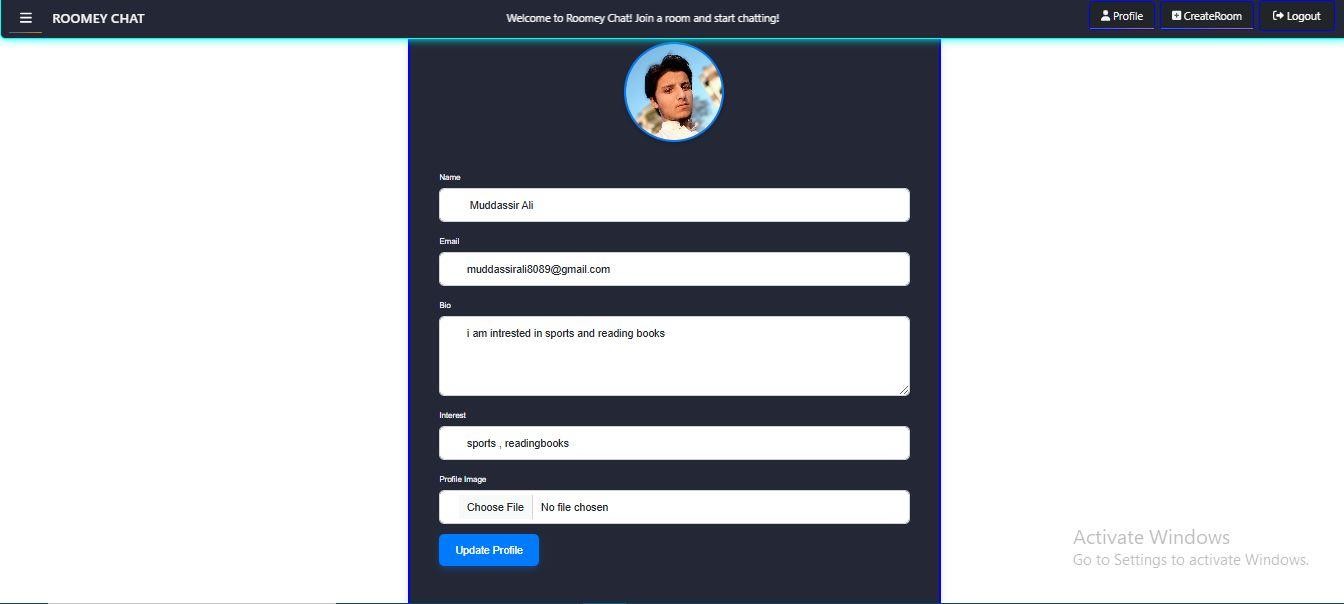
|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
| Zegocloud. | API integration platform that allows developers to add real-time communication, video calling, and live streaming features to their  applications | I am using the Api for video calling feature | Video.js |

## User Interface

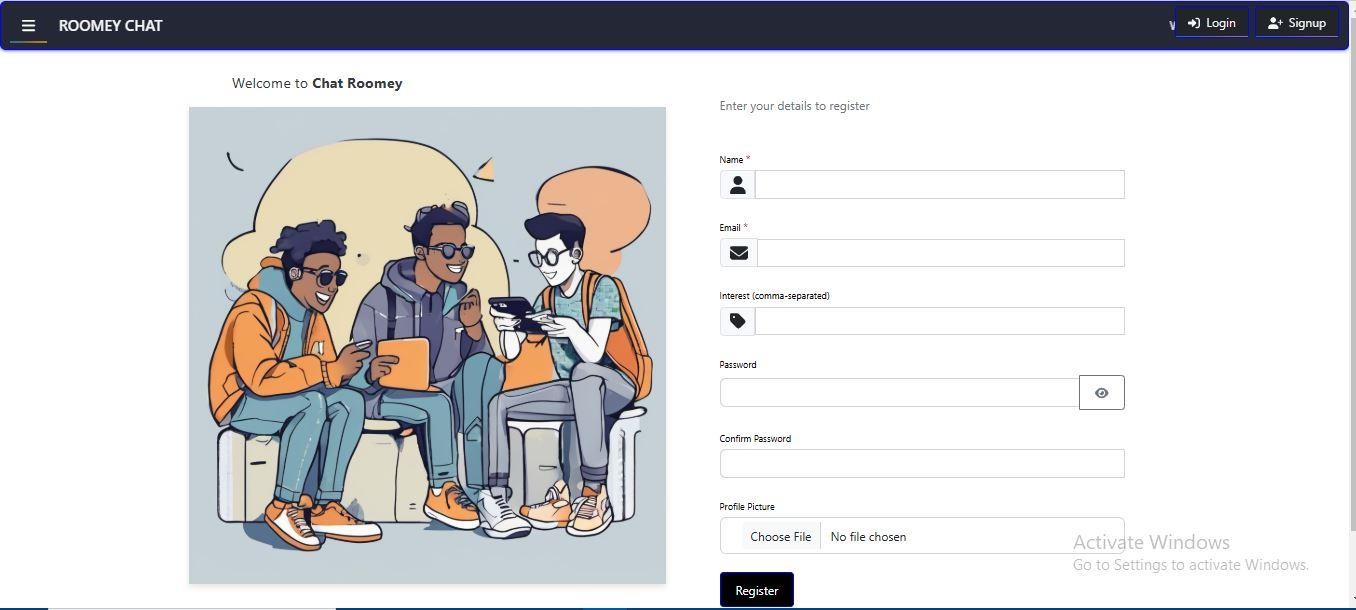
Details about user interface with descriptions.



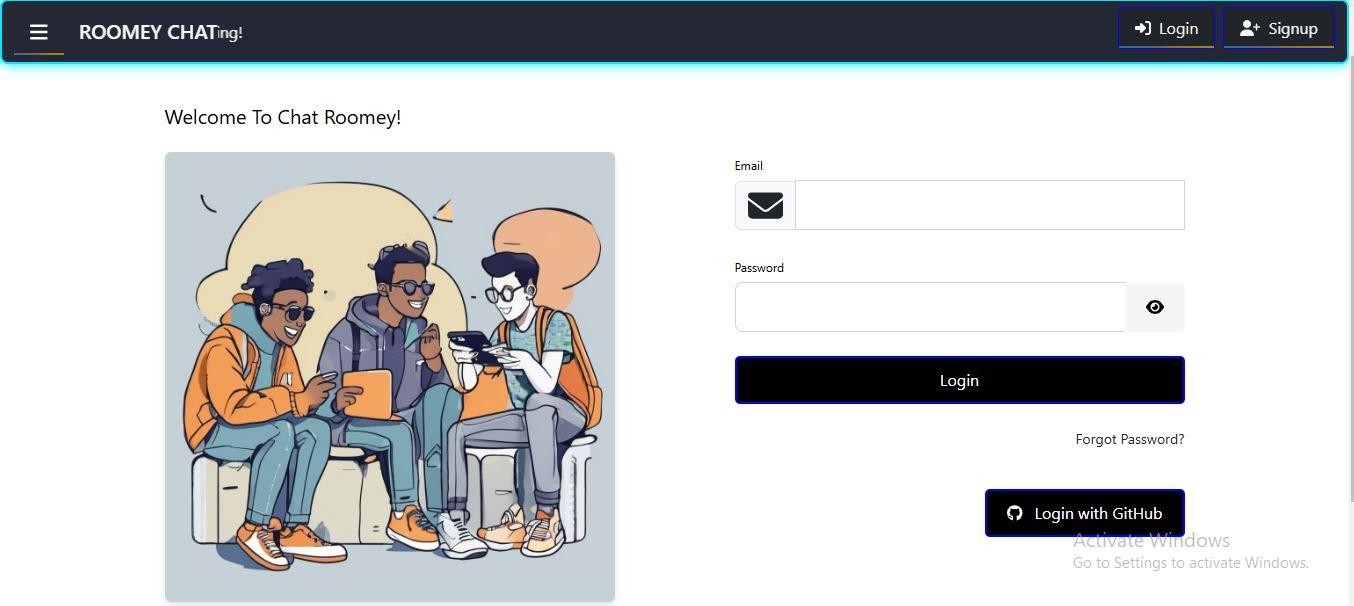
*Fig 5.1 Home Page*



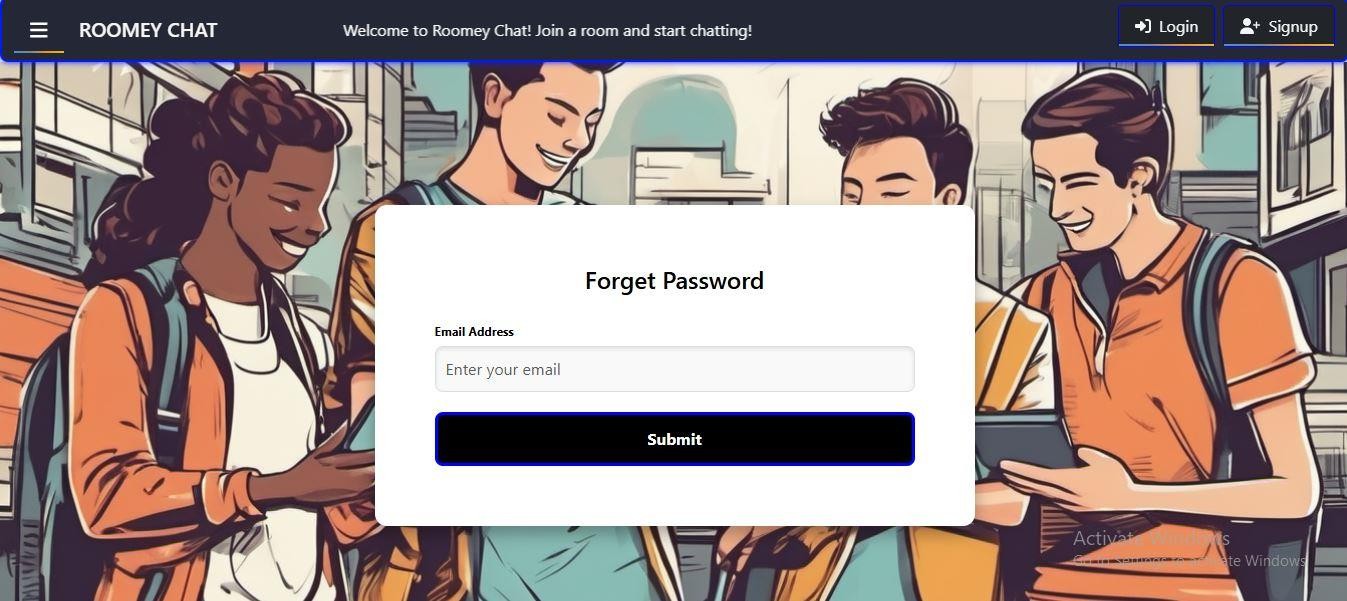
*Fig 5.2 Profile page*

**

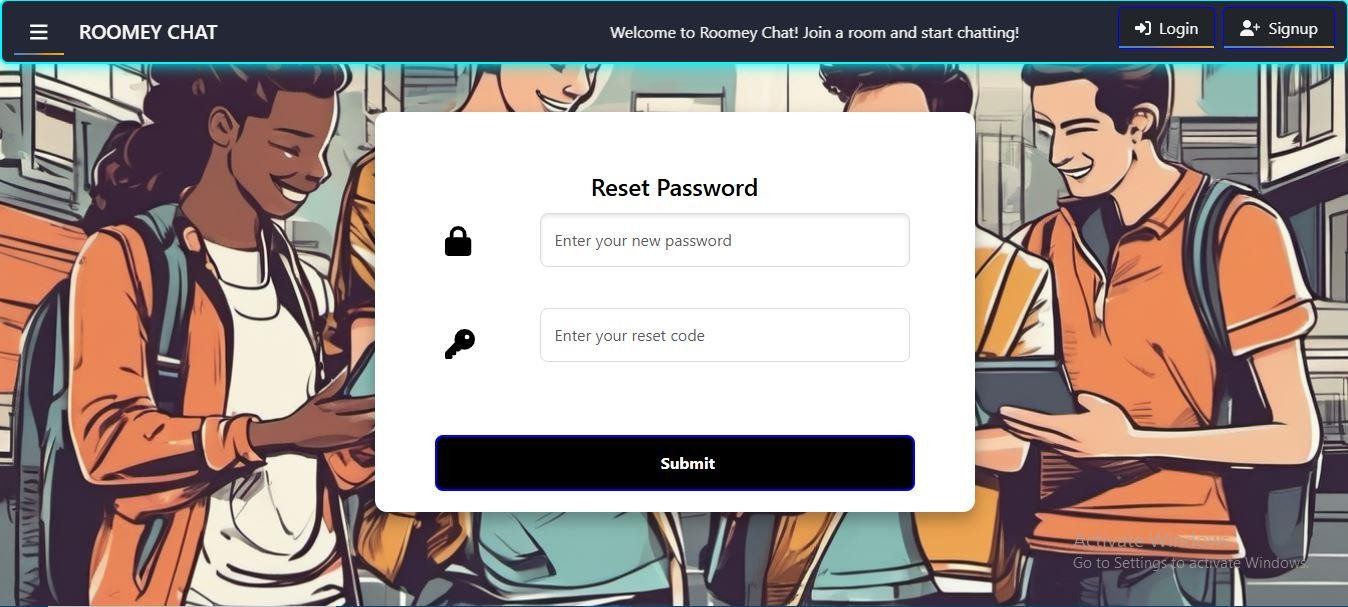
*Fig 5.3 Registration page*



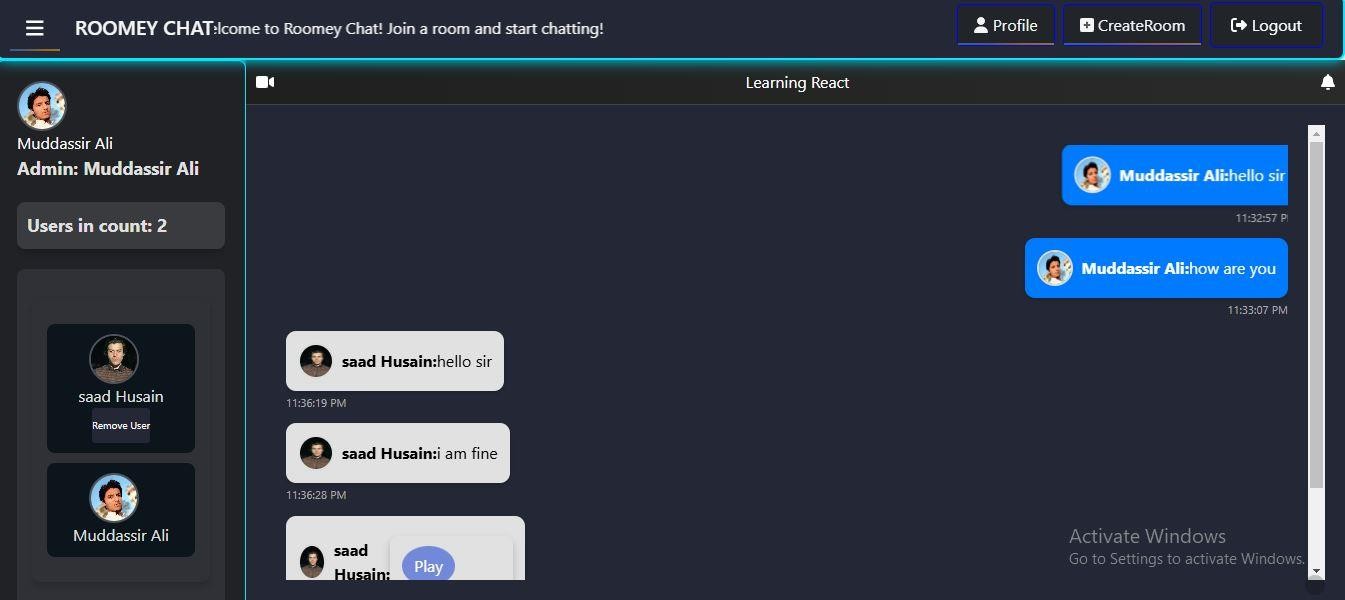
*Fig 5.4 Login page*

**

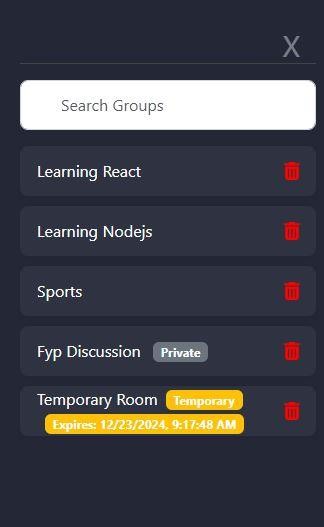
*Fig 5.5 Forget password screen*



*Fig 5.6 Reset password screen*

**

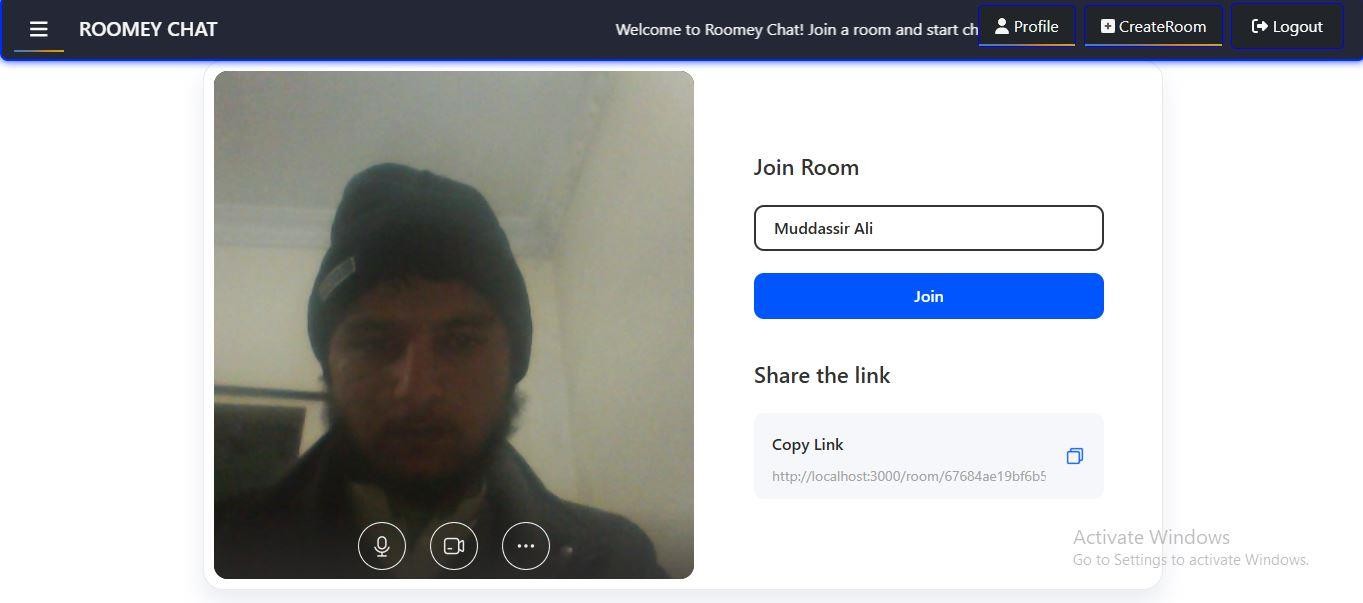
*Fig 5.7 messages page*



*Fig 5.8 List of Rooms page*



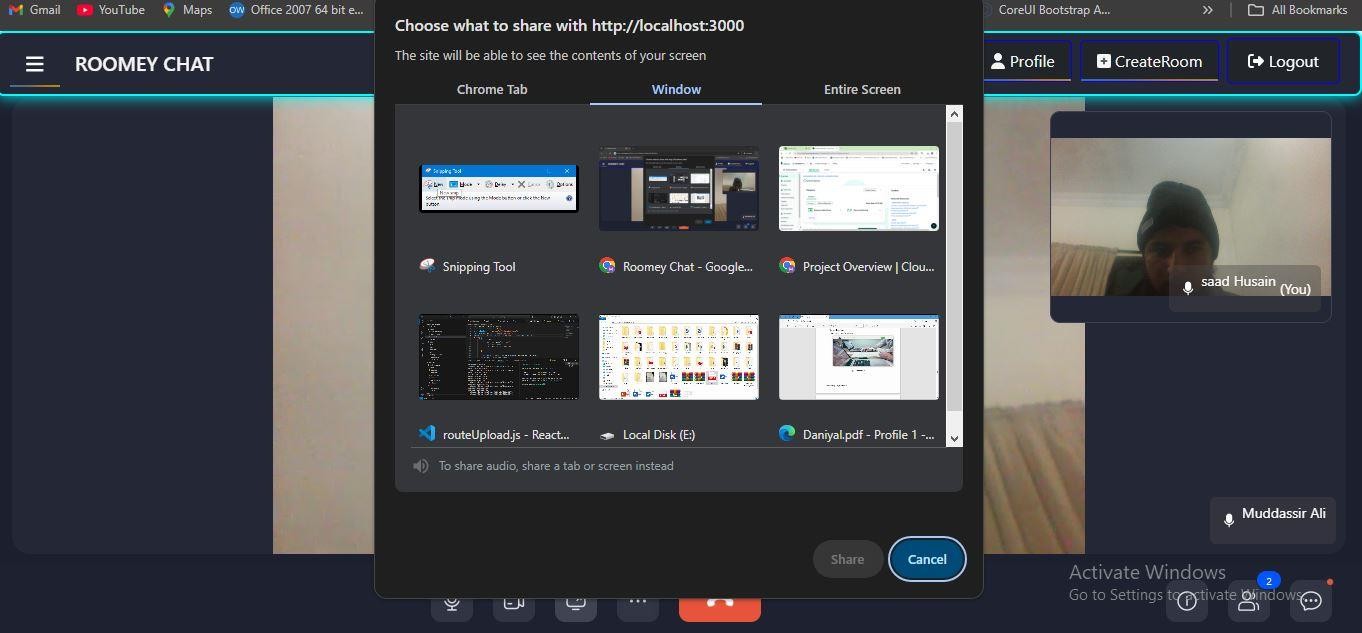
*Fig 5.9 List of users in group*



*Fig 5.10 Video call joining screen*

**

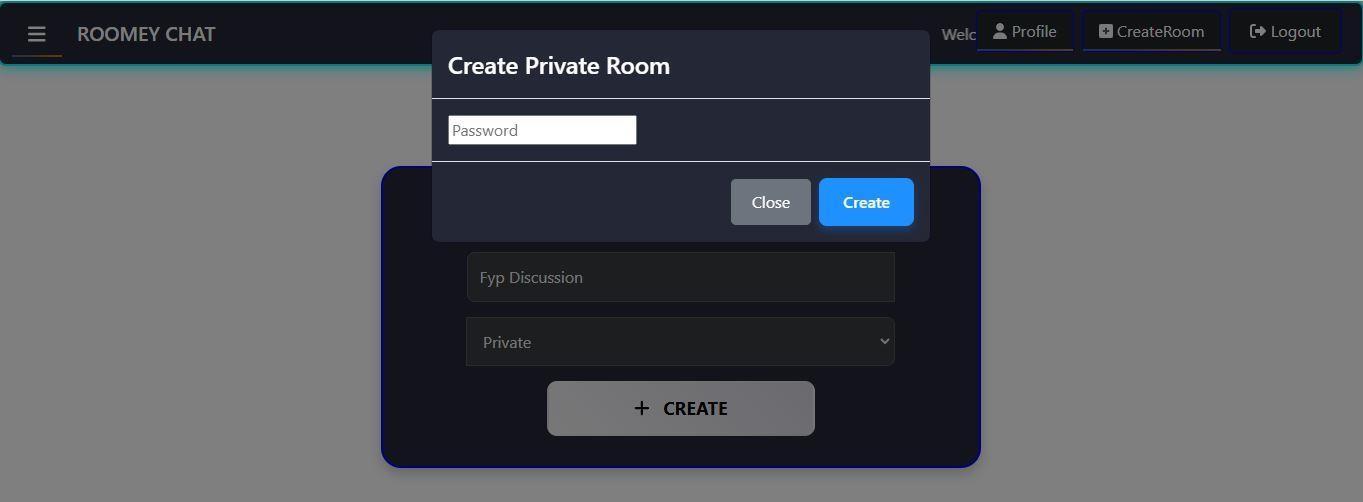
*Fig 5.11 Video calling screen*



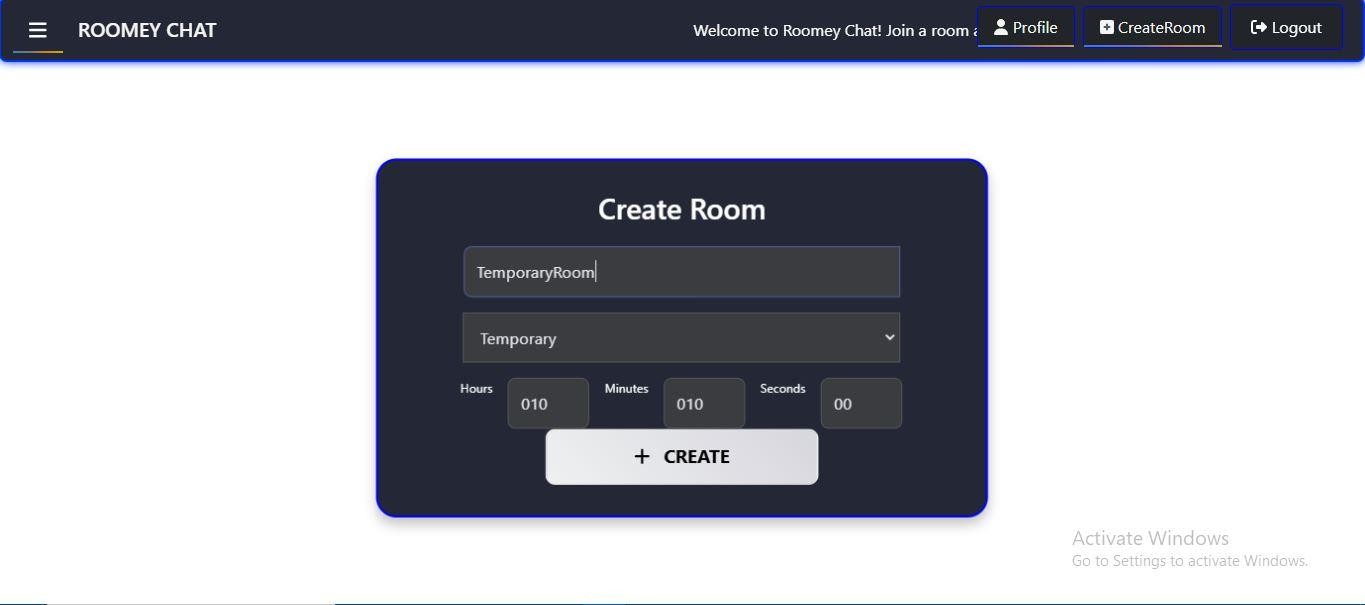
*Fig 5.12 Screen sharing window*

**

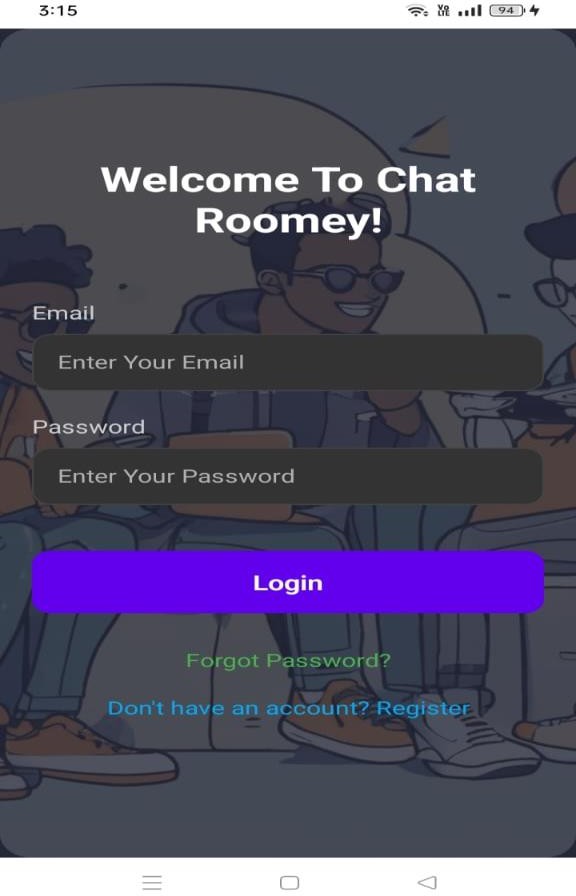
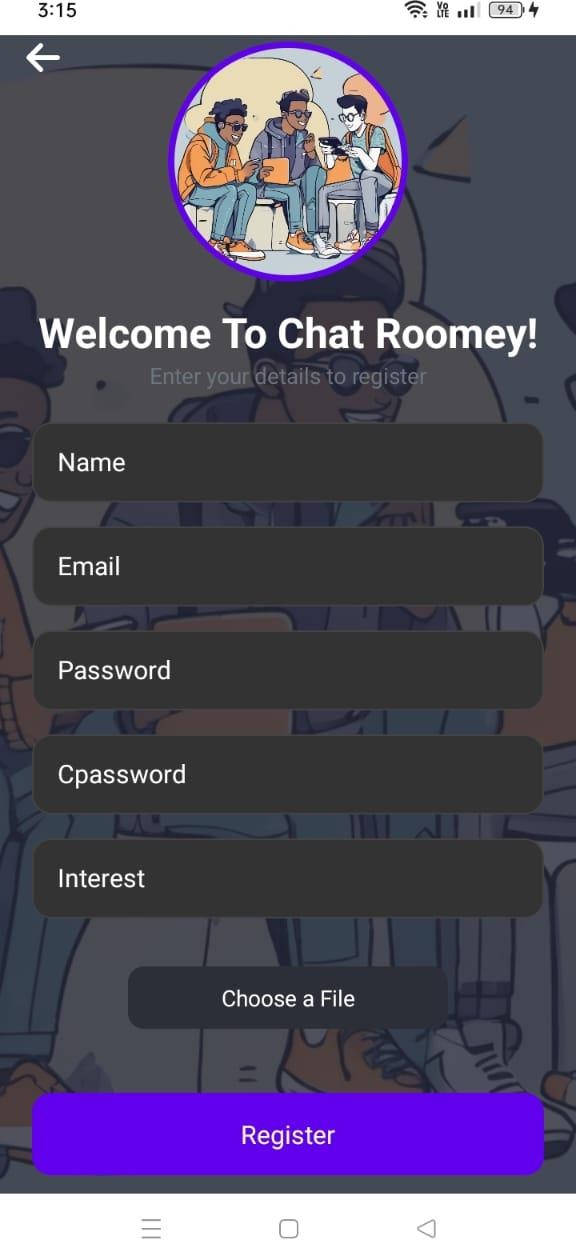
*Fig 5.13 Messaging in video call screen.*



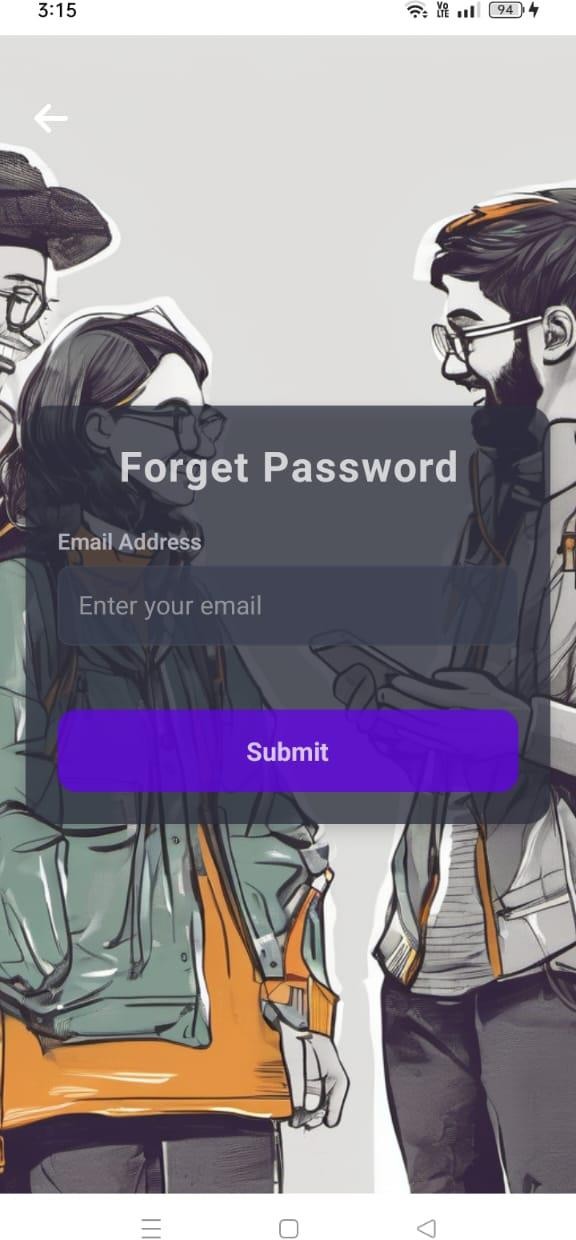
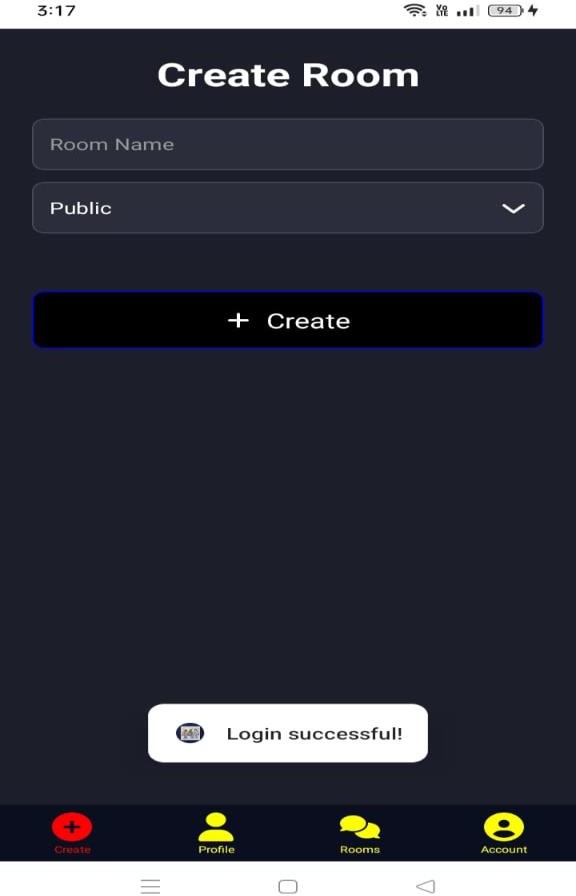
*Fig 5.14 Private Room screen*

**

*Fig 5.15 Temporary Room Screen*

*Fig 5.16 Login Screen Fig 5.17 Registration Screen*

*Fig 5.18 Forget password Screen Fig 5.19 Create Room Screen*

*Fig 5.20 Profile Management Screen Fig 5.21 Profile Management Screen*

# Testing and Evaluation

During software development, testing ensures that each component functions correctly, while evaluation assesses the software's overall performance and user satisfaction. Various types of testing, such as unit and integration testing, verify functionality, while usability testing and performance metrics gauge user experience and system efficiency

## Manual Testing

It ensures that software functions as expected from a user's perspective, identifying user interface issues, usability concerns, and functional bugs that might be missed by automated tests

### System testing

This table outlines the various aspects and stages of system testing conducted in the software development lifecycle.

*Table 6.1: System Testing Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Testcase/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| 1 | User Login - Valid Credentials | Email:  [muddassirali8089@gmail.com](mailto:muddassirali8089@gmail.com) Password: alsdfhadsf | Successfully log onto the main page of Roomy | Pass |
| 2 | Invalid Login Attempt | Email: [wrong@gmail.com](mailto:wrong@gmail.com)  Password: wrong password | Display error message incorrect username or password. | pass |
| 3 | Create Chat Room | Room name :Sports Select type : permanent | New chat room created and display in the list | Pass |
| 4 | Create Temporary Chat Room | User logged in. Room name : DSA  Select type : temporary  Enter time 10hours 10 min 5 sec | Temporary Room will be created and show in the list | Pass |
| 5 | Auto delete temporary chat room | Completion of the time | Room will automatically deleted and deleted from list | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Schedule Chat Room | Select the schedule date using calendar  19 dec 2024 | New schedule room created and display in the list | pass |
| 7 | Private Chat Room | User logged in. Room name : DSA Select type : Private  password:”1234” | New private room created and display in the list | pass |
| 8 | Join private room | Valid password | Private room join successfully | pass |
| 9 | Send Message in Chat Room | User logged in, Room: "Sports", Message: "Hello everyone" | Message sent and displayed in chat room. | Pass |
| 10 | Join Chat Room | User logged in, click on the room select the room | Successfully join the chat room. | Pass |
| 11 | Invite User to Chat Room | User logged in, and click on copy link button and send it the user | Invitation sent to the user, and they can join the room. | Pass |
| 12 | Search Chat Rooms | User logged in Search : "sports" | Display a list of rooms matching "sports". | Pass |
| 13 | Profile Customization | User logged in, Update interests and profile picture | Profile updated successfully with changes reflected on the profile page. | Pass |
| 14 | Video Call in Room | User logged in,  Click on the video icon and gave room Id video call is starting and other user click on join room | Successfully start and end the call with video enabled. | Pass |
| 15 | Room Deletion by Creator | User logged in, Room: "sports", Creator of room | Room deleted successfully. | Pass |
| 16 | User Logout | User logged in | Successfully log out and | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | return to the login screen. |  |
| 17 | Notification for New Message | User logged in, Room: "sports",  New message: "Welcome" | User receives a notification for a new message in the chat room. | Pass |
| 18 | Multiple Users in Room | Multiple users logged in, Room: “sports" | Multiple users can join and participate in the room without issues. | Pass |
| 19 | Display the list of join users in the room | User login and go to the group page | Group Join person names display in the list | Pass |

### Unit Testing

1. **User Authentication Module:**
   * **Testing Objective:** To ensure the edit profile form is working

*Table 6.2: Login Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Registration Valid Input | Username: Muddassir Ali Email:  [muddassirali8089@gmail.co](mailto:muddassirali8089@gmail.com) [m](mailto:muddassirali8089@gmail.com)  Password: 123Abc@  confirm password: 123Abc@ | Registration successful and user navigate to login page | pass |
| UT- 002. | After selecting the "Login" button on the login form and entering the relevant information, confirm the user's login. | Email: [muddassirali8089@gmail.co](mailto:muddassirali8089@gmail.com) [m](mailto:muddassirali8089@gmail.com)  Password: 123Abc@ | successfully access the system roomy's main page | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UT- 003. | Invalid Login Attempt | Email: [wrong@gmail.com](mailto:wrong@gmail.com)  Password: wrong password | Display error: "Incorrect username or  password" | Pass |
| UT- 004. | User Login - Empty Fields | Email:“” Password: “” | Display error: "Fields cannot  be empty" | Pass |

### Profile Management Module

* + **Testing Objective:** To ensure the edit profile form is working properly

*Table 6.3: Profile Management Module Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Profile Customization  - Valid | Update name and upload profile picture | Profile updated successfully | Pass |
| UT- 002. | Profile Customization  - Invalid  Picture | Upload unsupported file format (e.g., .exe) | Only png , jpg format accepted | pass |
| UT- 003. | Profile Customization  - Empty Bio | Leave bio blank | Donot leave the bio expty | pass |

### Chat Room Creation Module

* + **Testing Objective:** To ensure that chatroom creation module works properly

*Table 6.4: Chat Room Creation Module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Create Chat Room - Valid | Room name: "Sports", Type: Permanent | Chat room created and displayed in the list | Pass |
| UT- 003. | Create Chat Room - Empty  Name | Room name: “”, Type: Permanent | Display error:  "Room name cannot be empty" | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UT- 004. | Schedule Chat Room | Select the schedule date using calendar  19 dec 2024 | New schedule  created and display in the list | pass |
| UT- 005 | Video Call in Room | User logged in,  Click on the video icon and gave room Id video call is starting and other user click on join room | Successfully start and end the call with video enabled. | Pass |

### Temporary Chat Room Module

* + **Testing Objective:** To ensure that the temporary chat room creation module works properly

*Table 6.5: temporary chat room creation Module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Create tempory Chat Room | Select: "temporary", Ending time  10 hours 10 min 10 sec | Rooms will we remove after the completion of time | pass |
| UT- 002. | Create Temporary Chat Room -  Invalid Time | Time: "0 hours" | Display error: "Invalid time duration" | Pass |
| UT- 003. | Auto Delete Temporary Chat Room | Completion of set time | Room automatically deleted from the  list | Pass |

### Invitation Management Module

* + **Testing Objective:** To ensure that the temporary chat room creation module working properly

*Table 6.6: invitation module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Inviting User to Chat Room | User logged in, and clicked on copy link button and send it the  user | Invitation sent to the user, and they can join the room. | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UT-  002. | Invite User -  Invalid Link | Use a corrupted or  incomplete link | Display error: "Invalid  invitation link | pass |

### Search Rooms module

* + **Testing Objective:** To ensure that the search room’s functionality works properly.

*Table 6.7: search module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Search Chat Rooms - Valid Input | Search term: “Sports” | Display list of rooms matching the term. | Pass |
| UT- 002. | Search Chat Rooms - Invalid Input | Search term: "wrong name" | Display message: "No rooms found | pass |

### join Chat Rooms Module

* + **Testing Objective:** To ensure that the join functionality is working properly.

*Table 6.8: join chat rooms module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT-  001. | Join Chat Room  - Valid | Select room:  "Sports" | Successfully join the  selected chat room | Pass |
| UT- 002. | Search Chat Rooms - Invalid  Input | Search term: "wrong name" | Display message: "No rooms found | pass |

### Room Scheduling

* + **Testing Objective:** To ensure that the room scheduling is working properly.

*Table 6.9: Room Scheduling module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UT- 001. | Schedule Chat Room - Valid | Schedule date: 19  Dec 2024 | New scheduled  room created and displayed | Pass |
| UT- 002. | Schedule Chat Room - Invalid  Date | Schedule date: "Past date" | Display error:  "Invalid date selected" | pass |

### Real-Time Communication Module

* + **Testing Objective:** To ensure that real-time communication is properly.

*Table 6.10:* real time communication *Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT- 001. | Send Message in Chat Room | Room: “Sports”,  Message: "Hello everyone" | Display error:  "Message cannot be empty | Pass |
| UT- 002. | Send Message - Empty Input | Room: “Sports”, Message: "" | Display error:  "Invalid date selected" | pass |
| UT- 003. | Video Call - Valid | Start and join video call | Successfully start and end the video  call | Pass |
| UT- 004. | Video Call -  Invalid Room ID | Room ID: "Wrong123" | Display error: "Room not found" | Pass |
| UT- 005. | Notification for New Message | Room: "Sports",  New message: "Welcome" | User receives notification | Pass |
| UT- 006. | Multiple Users in Room | Multiple users  join and  participate | Room handles multiple users  without issues | Pass |

### Private chat rooms accessibility module

* + **Testing Objective:** To ensure that the private rooms are accessible properly.

*Table 6.11: private rooms accessibility module Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| UT-  001. | Create private  chat rooms | Select rooms type | Private chat  rooms created | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Private  room name:” sports”  Password:”123” |  |  |
| UT-  002. | Join private chat  rooms | Enter valid  password:”1234” | Room joins  successfully | pass |

### Functional Testing

This step checks if each module works correctly and makes sure the system meets all the required features and goals.

### User Registration:

*Table 6.12: User Registration Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Register with Valid Details | Username: Muddassir Ali Email: muddassirali8089 @gmail.com Password: 123Abc@ Confirm Password: 123Abc@ | User successfully registers and navigates to login page | Pass |
| FT- 002 | Register with Invalid Email Format | Username: Muddassir Ali Email: muddassir@wron g  Password: 123Abc@ Confirm Password: 123Abc@ | Error: 'Invalid email format' | Pass |
| FT- 003 | Register with Empty Fields | Username:“” Email:“” Password:”” Confirm  Password: “” | Error: 'Fields cannot be empty' | Pass |
| FT- 004 | Register with Weak Password | Username: Muddassir Ali Email:”” muddassirali8089 @gmail.com Password: 123 Confirm Password: 123 | Error: 'Password is too weak' | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FT- 005 | Register with Existing Email | Username: Muddassir Ali Email: muddassirali8089 @gmail.com Password: 123Abc@ Confirm  Password:123Abc @ | Error: 'Email already register' | Pass |
| FT- 006 | Password and Confirm Password Mismatch | Password:123Abc @  Confirm Password:  1234Abc | Error: 'Passwords do not match' | Pass |

### User Login

*Table 6.13: User Login Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Enter your valid credentials to log in. | Email: [muddassirali8089@gmail.com](mailto:muddassirali8089@gmail.com) Password: 123Abc@ | Successfully logged into the main page | Pass |
| FT- 002 | Enter an invalid email  address to log in. | Email: [mu@wronggmail.com](mailto:mu@wronggmail.com) Password: 123Abc@ | Error: Invalid email format | Pass |
| FT- 003 | Enter an incorrect password to log in. | Email: [muddassirali8089@gmail.com](mailto:muddassirali8089@gmail.com) Password: wrongpassword | Error: 'Incorrect username or password' | Pass |
| FT- 004 | Login with Empty Fields | Email: “”  Password: ‘’” | Error: Fields cannot be empty | Pass |

### Profile management

*Table 6.14: User Profile management Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test**  **Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Update Profile with Valid  Details | Bio: “interest in coding” Profile Picture: Uploaded | Profile updated successfully | Pass |
| FT- 002 | Update Profile  with Empty Fields | Bio:“”  Profile Picture: Not uploaded | Error: Fields  cannot be empty | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FT- 003 | Upload Invalid Profile Picture  Format | Profile Picture: invalidfile.txt | Error: 'Invalid file format' | Pass |

### Invitation management

*Table 6.15: User invitation management Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Send group invitation to the user | Click on copy link and send it to the user | Copy link successfully and invitation send | Pass |
| UT- 002 | Accept invitation | Users click on the link and user join the group | User successfully join the room | Pass |

### User list in the group

*Table 6.16: User view list of users Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | View User List | Group: "Project A" | List of users displayed | Pass |

### Room creation

*Table 6.17: Room creation Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Create Room with Valid Details | Room Name: Sports Type: Public | Room created successfully | Pass |
| FT- 002 | Create Room with Empty Name | Room Name: "" | Error: Room name cannot be empty | Pass |

### Search Rooms

*Table 6.18: Search Rooms Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Search Rooms | User logged in  Search : "sports" | Display a list of rooms matching "sports". | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FT- 002 | Search Rooms | User logged in  Search : “” | Room not found | pass |

### Delete Rooms

*Table 6.19: Delete Rooms Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Room Deletion by Creator | User logged in, Room: "sports", Creator of room  Click on delete icon | Room deleted successfully. | Pass |

### Temporary Room creation

*Table 6.20: Temporary Room creation Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Create Temporary Room | Room Name: "Temporary Room" Duration: 2 hours 2  min 30 sec | Temporary room created successfully | Pass |
| FT- 002 | Auto Delete Temporary Room | Time expires for temporary room | Room automatically deleted | Pass |

### Private Room creation

*Table 6.21: Private Room creation Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Create Private Room | Room Name:  "Private Chat" Type private | Room created successfully | Pass |
| UT- 026 | Access with wrong password | Access without password | Error: ‘enter a valid password’ | Pass |

### Schedule Room

*Table 6.22: Schedule Rooms Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FT-001 | Schedule Room | Room Name: "Scheduled Room"  Date: 19 dec  2024 | Room scheduled successfully | Pass |
| FT-002 | Schedule Room with Past Date | Date: previous date | Error: "Invalid date" | Pass |

### Video calling

*Table 6.23: Video calling Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT-001 | Start Video Call | Click on "Start Call icon" | Video call started | Pass |
| FT-002 | Join Video Call | Click on "Start Call icon and  join " | Successfully joined the video  call | Pass |

### Messaging audio/text

*Table 6.24: Messaging audio/Text Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT-001 | Send Text  Message | Message:  “Hello” | Message sent  successfully | Pass |
| FT-002 | Send Audio Message | Record and send audio message | Audio message sent successfully | Pass |

### Logout functionality

*Table 6.25: Logout Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| FT- 001 | Logout from Main Page | Click on "Logout" | User logged out and redirected to login page | Pass |

### 6.1.3. Integration Testing

Table 6.23 shows the integration testing

*Table 6.26: Integration Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| IT- 001 | Successful Registration and Login | Registration details: Email: muddassirali80  [89@gmail.com](mailto:89@gmail.com) | Registration and login succeed, user | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | , Password: 123Abc@, Confirm Password:  123Abc@ | navigated to the dashboard |  |
| IT- 002 | Registration and Login  with Invalid Email | Email:  muddassirali@ | Registration  fails; error: | Pass |
|  |  | invalid, | 'Invalid email |  |
|  |  | Password: | format' |  |
|  |  | 123Abc@ |  |  |
| IT- 003 | Registration with Valid Details but Login with Wrong Password | Registration succeeds.  Login details: Email:  muddassirali80 | Login fails; error: 'Incorrect username or password' | Pass |
|  |  | [89@gmail.com](mailto:89@gmail.com) |  |  |
|  |  | Password: |  |  |
|  |  | wrong |  |  |
|  |  | password |  |  |

## Automated Testing:

Here is the automated test for Roomy

### Tools used:

Table 6.27 shows the automated testing.

*Table 6.27: automated testing*

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool Name** | **Tool Description** | **Applied on [list of related tests cases / FR / NFR]** | **Results** |
| Selenium WebDriver | A framework for automating web applications for testing purposes. | login functionality,  Form submissions, Navigation testing. | Pass |
| Postman | A collaboration  platform for API development. | API endpoints testing. | Pass |

# Conclusion and Future Work

This chapter concludes the project and highlights future work.

## Conclusion

In conclusion, the development of the Roomy platform provides a versatile solution to enhance communication, foster collaboration, and facilitate efficient interaction through text, audio, and video chat rooms. By utilizing modern technologies and following software engineering best practices, Roomy addresses the limitations of traditional communication platforms, delivering a user-friendly and feature-rich experience for individuals and groups. Throughout the development process, the project incorporated key concepts from various course modules, such as web development, database management, and secure authentication. Rigorous testing and evaluation ensured the system’s reliability, functionality, and security, establishing Roomy as a robust platform for dynamic and engaging online interactions

## Future Work

While Roomy offers an innovative and efficient platform for communication and collaboration, there are several avenues for future work and enhancements: Integrate real-time language translation features to support multilingual communication, making Roomy accessible to a global audience. Develop a dedicated mobile application to provide users with seamless access to Roomy’s features on-the-go, enhancing convenience and usability. Implement AI tools for real-time content moderation and user behavior analysis to ensure a safe and productive communication environment. Enable Roomy to support large-scale virtual conferences and events with features like attendee management and breakout rooms. Incorporate advanced scheduling options with third-party calendar integration to streamline event planning and notifications. Establish a feedback loop to gather user input and perform iterative system updates, ensuring Roomy evolves to meet changing user needs. By exploring these enhancements, Roomy can continue to grow as a leading communication platform, offering versatile solutions for individuals and organizations alike.

# References

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

1. John Smith, Building Chat Applications with WebSocket’s and Node.js, [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\_API,](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) Last date accessed: December 21, 2024.
2. Page Author, WebSocket Basics for Real-Time Applications, https://socket.io/docs, Last date accessed: December 21, 2024.
3. Pooja Sharma, An Overview of RESTful APIs in Chat Applications, (2021), International Journal of Software Development, Vol. 10, Issue 3, pp 50-55.

Brian Cooke, Scaling Real-Time Chat Systems, (2020), Real-Time Systems Engineering Journal, pp 120-130

71