****

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**Roomy**

***By***

**Saad Hussain CIIT/SP21-BSE-020/ATD**

**Muddassir Ali CIIT/SP21-BSE-016/ATD**

**Muhammad Haris CIIT/SP21-BSE-019/ATD**

***Supervisor*Fuzzel Jameel**

***Bachelor of Science in Computer Science (2021-2024)***

**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**.

**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 report entirely on the basis of 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.

Student Name 1 Muddassir Ali Student Name 2

--------------------------- ------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Project title” was developed by   
**STUDENT 1 NAME (CIIT/FAXX-BCS/SE/TN-000)** and **STUDENT 2 NAME (CIIT/FAXX-BCS/SE/TN-000)** under the supervision of “SUPERVISOR NAME” and co supervisor “CO-SUPERVISOR NAME” 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 dynamic web application created to help real-time communication and collaboration within themed chat rooms. In contrast to other messaging platforms, Roomy focuses on creating temporary chat rooms based on specific interests or topics, where users can join and have conversation in text, audio or video. The application uses MERN stack, with MongoDB for data storage, Express.js for backend API development, React.js for frontend user interface, and Node.js for server-side operations. Through Roomy, users can find and join chat rooms they share interest in, connect with like-minded people, and take part in discussions in a enjoyable and open environment.

Key Features of Roomy include:

Room Creation and Discovery: Roomy lets users to create new chat rooms based defined subjects, hobbies, or topics. provides a search feature for users to find existing rooms.

Real-Time Chat: It enables real-time text messaging within chat rooms, allow users to send and messages on the go. It uses features such as messages, emojis, and reactions to enhance communication.

Audio Chat Rooms: Create audio chat rooms where users can have real time voice chats. Provide features for joining and leaving audio channels, similar to voice chat platforms.

Room Scheduling: Allow users to schedule rooms for specific dates and times, ensuring that discussions occur at convenient times for participants. Implement a calendar interface for browsing upcoming room schedules.

User Profiles and Preferences: Provide users with customizable profiles where they can add interests, bio information, and profile pictures. Allow users to set notification preferences and privacy settings.

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

**ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Majid Iqbal Khan” and our Co-Supervisor “Mr. Mukhtar Azeem”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

M. Saad Hussain Muddassir Ali Muhammad Haris

--------------------------- --------------------- ----------------------

**ABBREVIATIONS**

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

**TABLE OF CONTENTS**

[**1.** **Introduction** 12](#_Toc180092202)

[**1.1.** **Brief Introduction** 12](#_Toc180092203)

[**1.2.** **Relevance to Course Modules** 12](#_Toc180092204)

[**1.3.** **Project Background** 13](#_Toc180092205)

[**1.4.** **Literature Review** 14](#_Toc180092206)

[**1.5** **Analysis from Literature Review (in the context of your project.** 14](#_Toc180092207)

[**1.6** **Methodology and Software Lifecycle for This Project** 15](#_Toc180092208)

[**1.6.1.** **Agile Software Development Methodology** 15](#_Toc180092216)

[**2.** **Problem Definition** 18](#_Toc180092217)

[**2.1.** **Problem Statement** 18](#_Toc180092222)

[**2.2.** **Deliverables and Development Requirements** 19](#_Toc180092223)

[**2.3.** **Current System (if applicable to your project)** 20](#_Toc180092224)

[**3.** **Requirement Analysis** 22](#_Toc180092225)

[**3.1.** **Use Case Diagram(s)** 22](#_Toc180092230)

[**3.2.** **Detailed Use Case** 22](#_Toc180092231)

[**3.3.** **Functional Requirements** 37](#_Toc180092232)

[**3.4.** **Non-Functional Requirements** 38](#_Toc180092233)

[**4.** **Design and Architecture** 41](#_Toc180092234)

[**4.1.** **System Architecture** 41](#_Toc180092243)

[**4.2.** **Data Representation [Diagram + Description]** 43](#_Toc180092244)

[**4.3.** **Process Flow/Representation** 51](#_Toc180092245)

[**4.4.** **Design Models [along with descriptions]** 53](#_Toc180092246)

[**5.** **Implementation** 56](#_Toc180092247)

[**5.1.** **Algorithm** 56](#_Toc180092249)

[**5.2.** **External APIs** 56](#_Toc180092250)

[**5.3.** **User Interface** 56](#_Toc180092251)

[**6.** **Testing and Evaluation** 57](#_Toc180092252)

[**6.1.** **Manual Testing** 57](#_Toc180092261)

[This is the sample text 57](#_Toc180092262)

[**6.1.1.** **System testing** 57](#_Toc180092270)

[**6.1.2.** **Unit Testing** 57](#_Toc180092271)

[**6.1.3.** **Functional Testing** 58](#_Toc180092272)

[**6.1.4.** **Integration Testing** 58](#_Toc180092273)

[Table 5.4 shows the integration testing 58](#_Toc180092274)

[**6.2.** **Automated Testing:** 59](#_Toc180092275)

[This is the sample text 59](#_Toc180092276)

[**6.2.1.** **Tools used:** 59](#_Toc180092277)

[Table 5.5 shows the 59](#_Toc180092278)

[**7.** **Conclusion and Future Work** 60](#_Toc180092279)

[**7.1.** **Conclusion** 60](#_Toc180092281)

[**7.2.** **Future Work** 60](#_Toc180092282)

[**8.** **References** 61](#_Toc180092283)

**LIST OF FIGURES**

Fig 3.1 Use Case Diagram 21

Figure 4.2.1 System Sequance Diagram 42

Figure 4.2.2 Sequence diagram for Registration 43

Figure 4.2.3 Sequence diagram for Sign in 44

Figure 4.2.4 Sequence diagram for Manage Profile 45

Figure 4.2.5 Sequence diagram for create Chat Rooms 46

Figure 4.2.6 Sequence diagram for invite members 47

Fig 4.2.7 Sequence diagram for join chat room 48

Fig 4.2.8 Sequence diagram for search Rooms 49

Fig 4.2.9 Sequence diagram for update room 50

Fig 4.2.10 Sequence diagram for delete chat room 51

Fig 4.3.1 Process Flow/Representation 52

Fig 4.4.1 Data Flow Diagram level 0 53

Fig 4.4.2 Data Flow Diagram level 1 54

Fig 4.4.3 Data Flow Diagram level 2 55

**LIST OF TABLES**

Table 3.1 Register ..22

Table 3.2 Sign In ..22

Table 3.3 Manage Profile ..22

Table 3.4 Manage Profile ..22

Table 3.5 Create Chat Room ..22

Table 3.6 Search Room ..22

Table 3.7 Join Room ..22

Table 3.8 Schedule Chat Room ..22

Table 3.9 Create Groups (open/private) ..22

Table 3.10 Leave Room ..22

Table 3.11 Delete chatrooms 9

Table 3.12 log Out ..22

1. **Introduction**

Roomy is a web application made to facilitate real-time chat within themed chat rooms. Unlike other messaging platforms, Roomy focuses on creating temporary chat rooms based on specific subjects or interests, it allows users to join and participate in text or audio discussions. With Roomy, users can discover and join rooms tailored to user interests, it helps connect with like-minded persons and take part in discussions in a fun and open environment.

* 1. **Brief Introduction**

The project focuses on developing *Roomy*, a web application facilitating real-time chat in a themed environment. The goal is to facilitate users with a platform to have discussions on specific interests, it promotes focused and dynamic communication compared to other messaging platforms.

**Outcome:**

The outcome of this project is a web application which will provide a wide range of features, that includes creating and searching themed chat rooms, text and audio chats, room scheduling, and moderation tools. This provides communication among users who share common interests.

**Tools and Methodology:**

The project is developed by using Agile methodology, through iterative development, which focuses on continuous user feedback, and flexibility to changes throughout the whole project lifecycle.

* 1. **Relevance to Course Modules**

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

* + 1. **Database Management:**

Database management systems knowledge is necessary for Roomy as it covers storing user information, chat history, profile management and room data.

Database design, normalization, and query optimization will be applied to make sure efficient data storage and retrieval.

* + 1. **Web Technologies:**

*Roomy* is developed in web development technologies which provide a good experience by creating flexible user interface with back-end functionality. React.js will be used to build front-end and Node.js and Express will be used to build back-end. To save user data NOSQL data base like MongoDB is used which complete the practical knowledge of full-stack development.

* + 1. **Software Engineering Principles:**

The project uses software engineering principles and practices through its lifecycle. Requirements gathering, system design, coding and testing all are applied to ensure that the product is robust and scalable.

* + 1. **Human Computer Interaction (HCI):**

The designing of an application needs to have a good interface and user friendly. We will apply design methodologies that focuses on user needs. It will be engaging and productive for the features like chat room search, profile management, and easy navigation.

* + 1. **Project Management:**

*Roomy* follows Agile methodology software development that focuses on iterative development and continuous user feedback. We will apply agile practices like sprint planning, daily scrum meeting, sprint reviews and project/scrum backlog that develop out project management skills.

* + 1. **Privacy and Security:**

Security and privacy are main aspects in *Roomy*, handling user data and communication data is more important. We will learn about secure coding, authentication techniques, and data encryption to forbid unauthorized access of user data and make sure it meets data protection protocols.

* + 1. **Quality Assurance and Software Testing:**

Testing is an important part of *Roomy*’s development. We will use various testing techniques that include unit testing, integration testing, and user acceptance testing, to validate the system's functionality, reliability, and performance.

* 1. **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. The project takes inspiration from the increasing demand for specialized online communities, where individuals can connect, collaborate, and communicate in a more organized and interest-driven environment.

* 1. **Literature Review**

**Slack:**  
Slack is a messaging platform primarily aimed at workplace collaboration. It organizes conversations into channels, which can be topic-specific or team-based, and allows for direct messaging between users. It integrates with numerous productivity tools and supports features like file sharing, video calls, and threaded conversations. Slack’s success lies in its ability to streamline communication within organizations by reducing reliance on email and offering real-time messaging in a highly organized environment. However, while Slack excels in professional settings, it lacks the casual, community-building aspect found in other platforms.

**Discord:**  
Originally designed for gamers, Discord has evolved into a versatile communication platform used by a wide range of communities. It allows users to create servers, which can house multiple channels for both text and voice communication. Discord’s strength lies in its flexibility, offering a space for communities to engage in real-time discussions on various topics. Its audio chat feature is particularly robust, enabling users to join and leave voice channels with ease. Discord’s user-friendly interface, combined with its strong emphasis on communities and social interactions, makes it an ideal platform for interest-based communication. However, it lacks some of the professional integration tools seen in platforms like Slack.

**Comparison to Roomy:**  
Both Slack and Discord inspire Roomy in various ways. Like Slack, Roomy seeks to provide structured communication, but it is more focused on interest-based, temporary chat rooms rather than long-term collaboration. Like Discord, Roomy emphasizes community building through text and audio communication, but it differentiates itself by offering room scheduling and temporary discussions, enhancing flexibility and spontaneity.

* 1. **Analysis from Literature Review (in the context of your project.**

The analysis from the literature review provides valuable insights into the limitations of existing communication platforms like Slack and Discord, while identifying the opportunities for improvement that Roomy seeks to address. Here's how the literature review analysis contributes to the Roomy project:

* + 1. **Identification of Weaknesses in Existing Platforms:**  
       The literature review highlights several limitations in current communication platforms like Slack and Discord. While Slack is excellent for professional environments, it lacks features for casual, interest-based interactions. Discord, while great for communities, tends to focus on long-term groups rather than temporary, spontaneous chat rooms. This analysis reveals a gap for a platform like *Roomy*, which provides temporary chat rooms focused on specific topics, offering a more dynamic and flexible user experience.
    2. **Comparison with Proposed Solutions:**  
       By comparing the limitations of existing platforms with the solutions proposed by *Roomy*, the literature review underscores how *Roomy* addresses these gaps. For example, while Slack organizes communication into work-related channels and Discord focuses on long-term servers, *Roomy* allows users to create and join temporary rooms around specific interests. Additionally, *Roomy* offers room scheduling and audio chat functionality, combining the best aspects of both platforms while introducing new, interest-driven features.
    3. **Emphasis on Advantages and Benefits:**  
       The analysis from the literature review highlights the key benefits that Roomy brings to the table. These include fostering dynamic, real-time conversations in temporary chat rooms, providing both text and audio options for communication, and creating spaces for short-term, focused discussions. By focusing on creating temporary rooms, Roomy offers users more flexibility in joining discussions that matter to them, promoting deeper engagement and community-building in niche interest areas.
    4. **Scope and Limitations Consideration:**  
       The literature review also considers the scope and limitations of existing platforms, such as their focus on long-term groups or work environments, and the challenges they face in supporting spontaneous, temporary discussions. This analysis informs the development of Roomy, ensuring that the platform is designed to fill the gap for short-term, interest-based interactions while acknowledging potential challenges, such as moderating temporary rooms and managing user participation effectively.
  1. **Methodology and Software Lifecycle for This Project**

For the development of Roomy, a structured methodology and software development lifecycle (SDLC) model are critical to ensuring that the project progresses efficiently, meets user requirements, and delivers a high-quality, functional product. The selected methodology for this project is Agile, as it offers the flexibility and iterative approach needed for dynamic web applications like Roomy.

1. 6. 1. **Agile Software Development Methodology**

Agile methodology is particularly suited for *Roomy* due to its emphasis on iterative development, collaboration, and adaptability. This approach ensures that the project remains flexible and responsive to user feedback and evolving requirements throughout its lifecycle.

**Key Characteristics:**

Characteristics of Agile Software Development Methodology

**Iterative Development:** *Roomy* will be developed in iterative cycles, where each sprint delivers a specific set of features such as room creation, real-time chat, or audio chat functionalities. Each iteration will be followed by testing and review to ensure quality and alignment with project goals.

**Continuous Feedback:** Regular feedback loops will be integrated into the process, where stakeholders and potential users can provide input on newly developed features. This allows for quick validation of requirements and immediate adjustments based on feedback, ensuring that *Roomy* meets user expectations.

**Implementation:** Agile will be implemented using the Scrum framework, featuring key practices like sprint planning, daily stand-ups, sprint reviews, and retrospectives. Each sprint will focus on delivering functional features, and the team will collaborate to adjust and improvements in each subsequent cycle. This iterative and collaborative approach ensures a robust and user-cantered development process for *Roomy*.

* + 1. **Software Development Lifecycle (SDLC):**

Here is the Software Development Lifecycle:

**Requirement Analysis:** Gather and analyse requirements from potential users, such as individuals looking to join interest-based chat rooms, moderators, and administrators. This phase defines the core features of *Roomy*, such as room creation, real-time chat, audio chat, and moderation tools.

**Design:** Create the system architecture, database schema, and user interface (UI) designs based on the requirements from the analysis phase. This includes planning how chat rooms will be created, how users will discover them, and the backend logic for real-time messaging. The design should ensure smooth integration of the frontend and backend, prioritizing ease of use and scalability.

**Development:** Implement the functionalities of Roomy using modern web technologies. The development phase will include coding the frontend with React.js, building the backend APIs with Node.js and Express.js, and setting up MongoDB for data storage. This phase also involves integrating real-time messaging, audio chat, and user profile customization features.

**Testing:** Conduct thorough testing to ensure the quality and performance of Roomy. This includes unit testing for individual components, integration testing to ensure the system works, and user acceptance testing (UAT) to gather feedback from potential users on features such as room scheduling, chat functionalities, and the user interface. Special attention will be given to ensuring the stability of real-time communications.

**Deployment:** Deploy the Roomy application to a staging environment for final testing and validation. This phase involves testing the app’s performance under real-world conditions and ensuring that it works across different devices and browsers. Once all issues are resolved, Roomy will be deployed to the production server for public use.

**Maintenance and Support:** Provide ongoing maintenance and support for Roomy, which includes fixing bugs, releasing updates, and adding new features based on user feedback. Continuous monitoring will be essential to ensure the system remains stable, secure, and responsive to evolving user needs and technological changes.

1. **Problem Definition**

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

4. 1. **Problem Statement**

The problem addressed by the *Roomy* project comes from the limitations of current communication platforms like Slack and Discord when it comes to creating temporary, topic-focused chat rooms. These platforms are designed more for long-term teams or communities, which leaves a gap for users who want flexible, short-term spaces for conversations based on specific interests. The main problems identified are:

* + 1. **Lack of Temporary Chat Rooms:**

Most platforms focus on permanent chat groups, making it hard for users to create and join temporary rooms for short-term discussions about specific topics. This makes it difficult for users to have spontaneous, interest-driven conversations.

* + 1. **Limited Audio Chat for Short-Term Conversations:**

While Discord offers audio chat, it is tied to permanent servers. There aren’t many options for users who want to join temporary voice chats for real-time discussions on specific topics.

* + 1. **Difficult Room Discovery:**

On existing platforms, finding rooms that match a user's specific interests can be challenging, especially without joining large, established groups. A more user-friendly search and discovery feature is needed to make this easier.

* + 1. **No Room Scheduling Options:**

Most platforms don’t allow users to schedule rooms for specific times, which means discussions can be missed if not organized ahead of time.

* + 1. **Limited User Customization and Privacy Options:**

Current platforms don’t offer enough tools for users to personalize their profiles, set privacy preferences, or customize notifications to their liking. This reduces user satisfaction, especially for those who want a more tailored experience.

* + 1. **Challenges with Moderation:**

Moderating and managing chat rooms, especially temporary ones, can be difficult. Current platforms do not have strong tools for managing these kinds of chat spaces and ensuring rules are followed.

To address these issues, *Roomy* will create a platform that supports temporary, topic-based chat rooms with real-time text and audio options, easy room discovery, and scheduling. It will also provide strong moderation tools and customization options to improve the user experience.

* 1. **Deliverables and Development Requirements**

The Deliverables for the Roomy.

* + 1. **Fully Functional Web Application:**

Develop a comprehensive, user-friendly web application that allows users to create, join, and manage chat rooms. The application will support both temporary and permanent rooms, offering a flexible platform for communication**.**

* + 1. **User Registration and Login:**

Implement secure user registration and login systems to ensure users can create accounts, log in, and manage their profiles. This will include basic security measures like password encryption and account recovery options.

* + 1. **Profile Management:**

Provide users with the ability to customize and update their profiles. This will include options to upload profile pictures, add bio information, set notification preferences, and adjust privacy settings.

* + 1. **Chat Room Creation and Management:**

Allow users to create their own chat rooms based on specific interests or topics. Users will have the ability to invite others, manage room settings, and decide whether the room is public or private.

* + 1. **Temporary and Scheduled Chat Rooms:**

Introduce features for creating temporary chat rooms that disappear after use, as well as scheduling rooms for future dates. This will include a calendar interface for organizing and scheduling discussions.

* + 1. **Real-Time Communication (Text, Audio, and Video):**

Implement real-time communication features, including text messaging, audio chat, and video calls within chat rooms. Users will be able to participate in discussions using their preferred communication method.

* + 1. **Group Creation (Open and Private):**

Enable users to create and manage both open and private groups. This will allow people with shared interests to collaborate and engage in discussions in a controlled environment.

* + 1. **Search Chat Rooms:**

Implement a search feature that allows users to quickly find chat rooms based on topics, interests, or specific keywords. This will enhance the user experience by making it easier to discover relevant rooms.

* + 1. **Delete Chat Rooms:**

Provide the ability for room creators or administrators to delete chat rooms when they are no longer needed, ensuring that the platform remains organized and clutter-free.

* + 1. **Secure and Scalable System:**

Ensure the application is secure with proper authentication and data encryption. The system will be built to scale, allowing it to handle increasing numbers of users and chat rooms as the platform grows.

* + 1. **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.

* 1. **Current System (if applicable to your project)**

At present, various platforms support real-time communication, each with unique features. Two of the most popular systems are **Slack** and **Discord**, which facilitate team collaboration and community-building through chat rooms or channels.

1. **Slack:**

Slack is primarily used in professional environments. It allows users to create channels for different projects or teams, where members can send text messages, share files, and integrate third-party tools like Google Drive, Trello, or GitHub. Slack is designed for businesses and organizations, making collaboration within teams easier. However, its free version limits the number of message searches and integrations, which may hinder its utility for smaller or short-term projects.

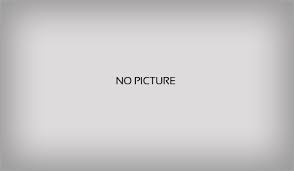
1. **Discord:**

Discord initially gained popularity in the gaming community but has since expanded to various online communities. Discord supports both text and voice communication within organized channels. It provides a user-friendly interface with options to create public or private servers where users can interact. While Discord is excellent for voice communication and community interaction, its emphasis on voice chat over text-based communication can be less useful for professional or specific discussion-based contexts.

**Limitations of Current Systems:**  
While both Slack and Discord are effective in fostering communication, they come with limitations that Roomy seeks to address. Slack's restricted free tier limits functionality for smaller groups, and its focus is on long-term workspaces rather than temporary chat rooms. Discord, on the other hand, is not well-suited for professional environments due to its focus on voice chats and gaming-centric design. Additionally, neither platform provides a streamlined way to create and schedule temporary chat rooms for specific discussions or events, which is essential for certain user groups.

**Roomy's Improvement:**  
Roomy is designed to overcome these limitations by offering a flexible platform where users can create **temporary chat rooms**, schedule discussions through a built-in **calendar**, and engage in **text, audio, or video communication** seamlessly. Roomy is suited for both professional and casual environments, providing a better solution for users who need temporary, interest-based discussions with enhanced control over room management.

The following figure is a sample figure, Figure 2.1. You are required to follow the same style of numbering and caption for the whole report.



*Figure 2.1: Sample picture*

The following table (Table 2.1) is sample table; you are required to follow the same style of numbering and caption for the whole report.

*Table 2.1: Sample Table*

|  |  |  |
| --- | --- | --- |
| **Header 1** | **Header 2** | **Header 3** |
| Text | Text | Text |
|  |  |  |

The following list style is the sample to consistently follow in the whole report.

* List items 1
* List items 2

1. **Requirement Analysis**

Here is a Use case Diagram of Roomy.

4. 1. **Use Case Diagram(s)**

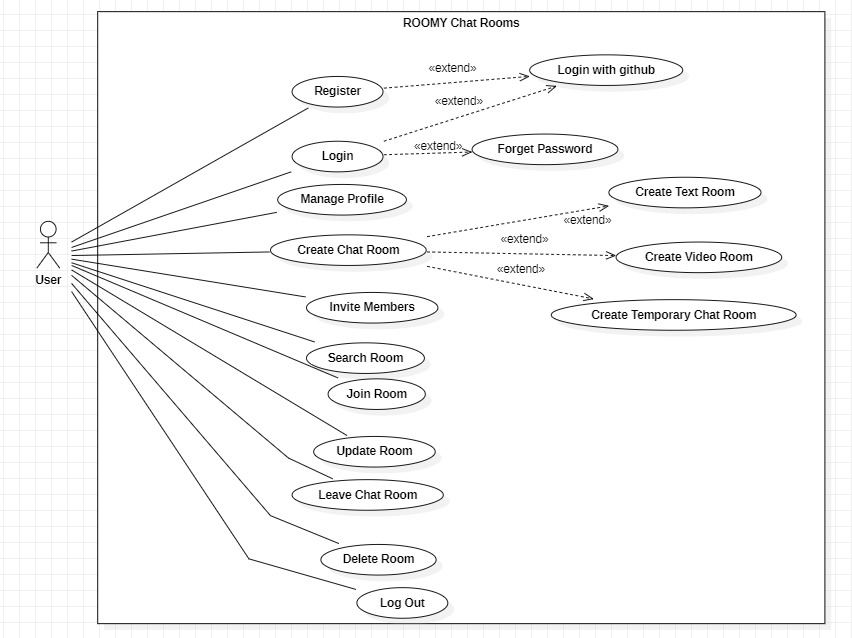
****

Fig 3.1 Use Case Diagram

* 1. **Detailed Use Case**

Here is the Detailed Use Case Diagram of Roomy.

* + 1. **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 want 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 Clicks on register button.  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.  The system asks the user to choose another username.  The user chooses another username.  The system then saves the username against the user. |
| **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 must contain 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. |

* + 1. **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 internet access.  The user has a valid email address. |
| **Postconditions** | The user |
| **Normal Flow** | The user opens Roomy app homepage.  The user click on the “Sign-In” button.  The User enters its email and password.  The user click on the login button.  The user email and password is validated.  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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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 manage 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.  The user is logged in to the 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, an error message is displayed, User is asked to try again later. |
| **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). |

* + 1. **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**  **Alternative flow 2**  **(Room Name Already Taken)** | The name provided by the user is already taken.  The system shows an error message to the user showing the name is already taken.  The user re- enters the name and resubmit it. |
| **Exceptions** | In case of system overload, an error message is displayed, User is asked to try again later. |
| **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). |

* + 1. **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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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 user clicks on the search button.  The system fetch the specific chat room from data base.  The user select 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 criteria and submits the search query.  The system processes the search request but finds no chat rooms matching the criteria.  The system displays a message indicating no matching chat rooms were found and suggests alternative search terms or criteria. |
| **Exceptions** | In case of system overload, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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.  If the room is private the user is asked to enter the password.  The user enters password.  The system evaluates the password.  If it enters the right password, the system allow 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 an invalid password.  The system shows an error message indicating the invalid 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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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 shows an error message to the user and asks the user to select a valid date and time |
| **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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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 use has access to the internet.  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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **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 is part of.  The user clicks on the "Leave Room" button.  The system asks the user to confirm the action.  The user confirms to leave the chat room.  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 prompts 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, an error message is displayed, User is asked to try again later. |
| **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. |

* + 1. **Use Case Description for Delete chat Rooms**

Table 3.11 Delete chatrooms

|  |  |
| --- | --- |
| **Use case ID** | UC -11 |
| **Use case Name** | Delete chatrooms |
| **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, an error message is displayed, User is asked to try again later.  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. |

* + 1. **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 is successfully logged into the Roomy app.  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 prompts 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, an error message is displayed, User is asked to try again later. |
| **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. |

* 1. **Functional Requirements**

The functional requirements of Roomy shows 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:

* + 1. **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.

* + 1. **Profile Management:**

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

* + 1. **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.

* + 1. **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.

* + 1. **Invitation Management:**

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

* + 1. **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.

* + 1. **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.

* + 1. **Real Time Communication**

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

* + 1. **Log Out and Account Security:**

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

* 1. **Non-Functional Requirements**

functional 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:

* + 1. **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**.**
    1. **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
  + 1. **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.
  + 1. **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.
  + 1. **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 a consistent experience on each.
  + 1. **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.
  + 1. **Interoperability**
* **Integration:** Roomy should work well with other systems and services, like calendars, external communication platforms, or APIs, for added features.
  + 1. **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**.**

1. **Design and Architecture**

Here is detailed Design and Architecture of Roomy.

8. 1. **System Architecture**

Here is the System Architecture of Romy:

* + 1. **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 frontend 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.

* + 1. **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.

.

* + 1. **Authentication Service**

Description: This module handles user authentication and access control within the Roomy 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.

.

* + 1. **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.

.

* + 1. **Communication Service**

Description: This module facilitates real-time communication between users in Roomy. It supports text messaging, audio, and video communications 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.

* + 1. **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.

.

* + 1. **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.

* + 1. **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. It is optimized for scalability, ensuring that as the user base grows, the database can handle increasing amounts of data. The MongoDB database is integrated with the backend to provide fast access to real-time data across all aspects of the platform.

* 1. **Data Representation [Diagram + Description]**

No table of contents entries found.

Here is detailed Design and Architecture of Roomy.

* + 1. **System Sequence Diagram**

**SSD here**

* + 1. **Sequence Diagram for Registration**

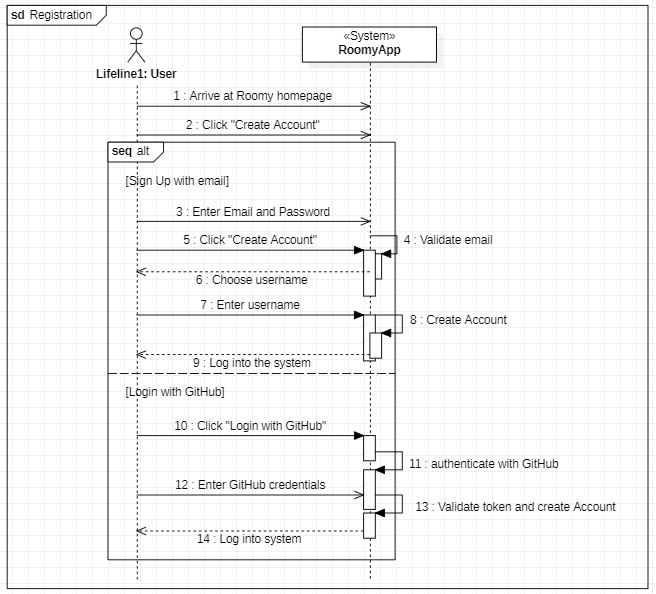


Figure 4.2.2: Sequence diagram for Registration

* + 1. **Sequence Diagram for Sign in**

A screenshot of a computer screen

Description automatically generated

Figure 4.2.3: Sequence diagram for Sign in

* + 1. **Sequence Diagram for Manage Profile**

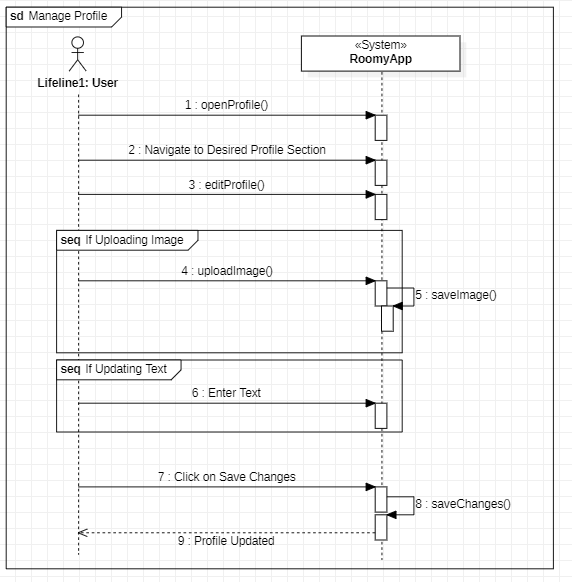


Figure 4.2.4: Sequence diagram for Manage Profile

* + 1. **Sequence Diagram for Create Chat Rooms**

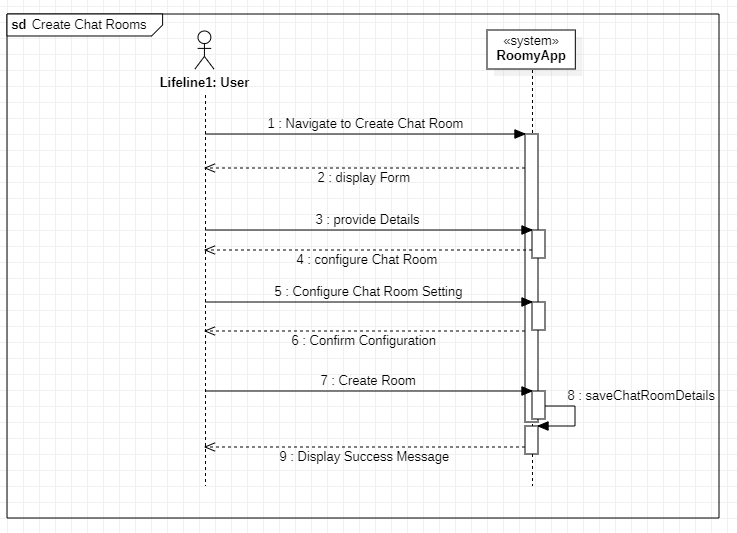


Figure 4.2.5: Sequence diagram for create Chat Rooms

* + 1. **Sequence Diagram Invite members**

**A screenshot of a diagram

Description automatically generated**

Figure 4.2.6: Sequence diagram for invite members

* + 1. **Sequence Diagram for join chat Room**

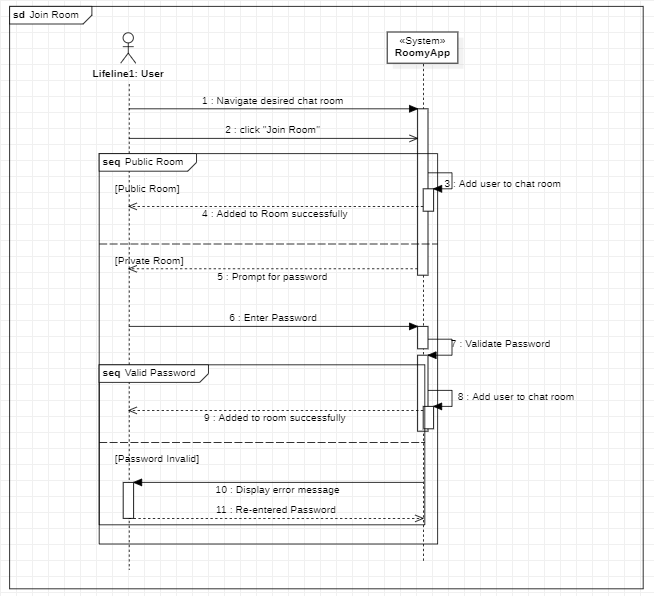


Figure 4.2.7: Sequence diagram for join chat room

* + 1. **Sequence Diagram for Search Rooms**

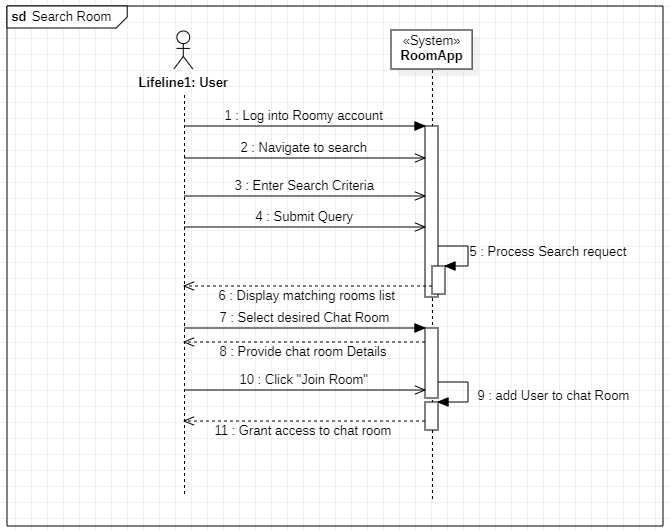


Figure 4.2.8: Sequence diagram for search Rooms

* + 1. **Sequence Diagram for update Room**

A diagram of a system

Description automatically generated

Figure 4.2.9: Sequence diagram for update room

* + 1. **Sequence Diagram for Delete Chat Room**

**A screenshot of a chat room

Description automatically generated**

Figure 4.2.10: Sequence diagram for delete chat room

* 1. **Process Flow/Representation**

Here is a process flow diagram for Roomy.

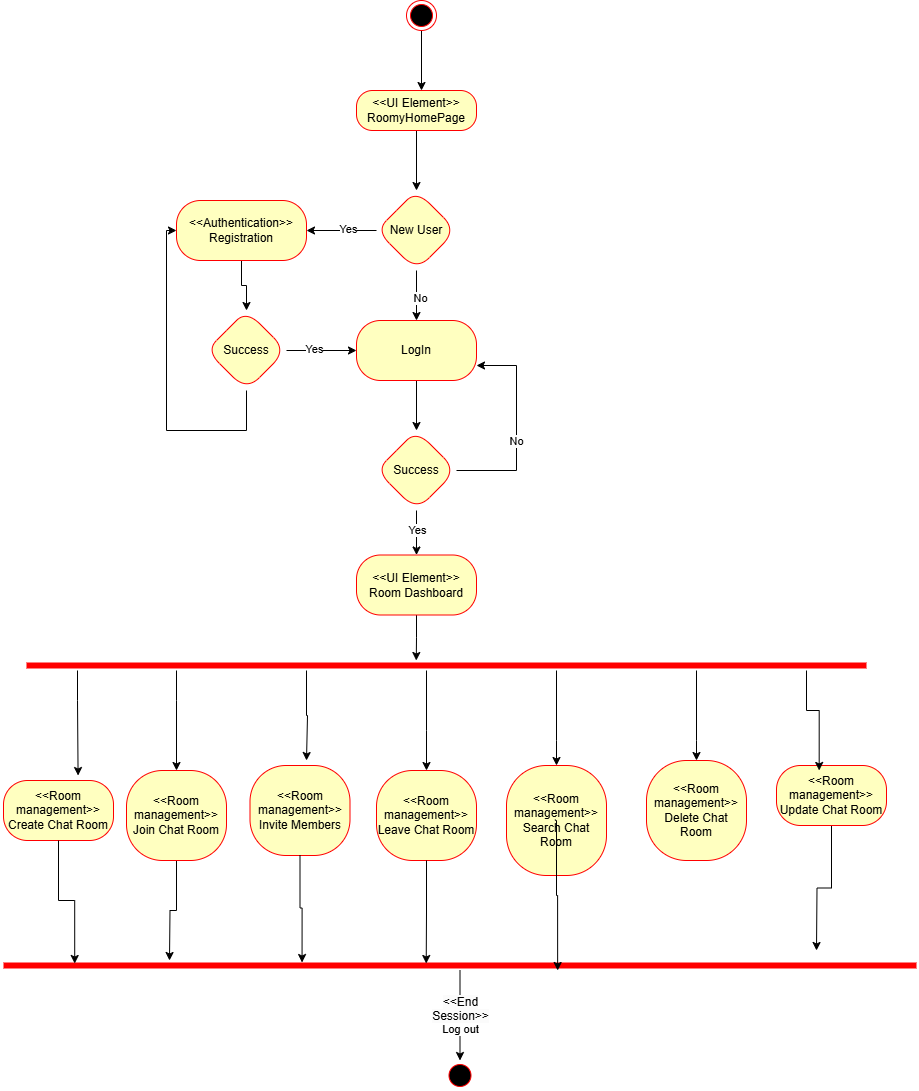


Fig 4.3 process flow diagram

* 1. **Design Models [along with descriptions****]**

Here is Design Models with different levels of diagram

* + 1. **Data Flow Diagram Level 0:**

A diagram of a roomy system

Description automatically generated

Figure 4.4.1: Data Flow Diagram level 0

* + 1. **Data Flow Diagram Level 1:**

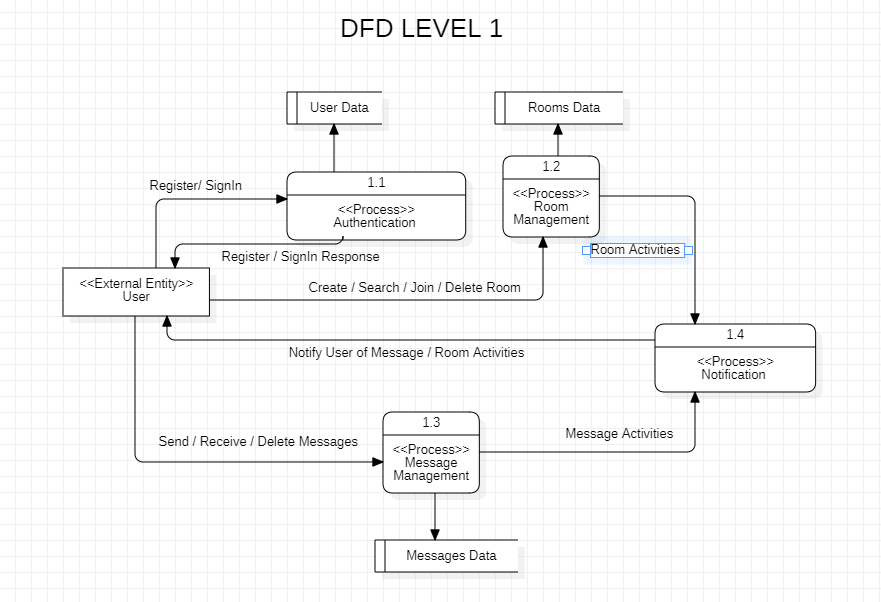


Figure 4.4.2: Data Flow Diagram level 1

* + 1. **Data Flow Diagram Level 2:**

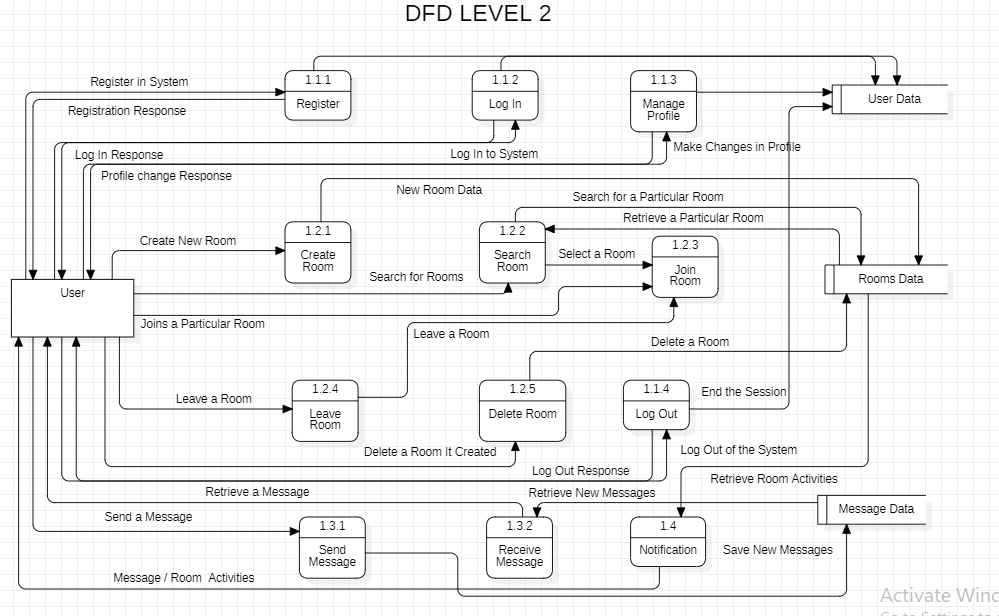
****

Figure 4.4.3: Data Flow Diagram level 2

1. **Implementation**

This chapter will discuss implementation details supported by UML diagrams (if applicable). You will not put your source code here. Any of the following sections may be included based on your project.

1. 1. **Algorithm**

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode **OR** a natural language explanation regarding the functioning of main features. Be sure to use the correct syntax and semantics for algorithm representations.

* 1. **External APIs**

Describe the APIs used in the table 5.1.

Table 5.1 shows that

*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** |
|  |  |  |  |
|  |  |  |  |

* 1. **User Interface**

Details about user interface with descriptions.

1. **Testing and Evaluation**

This chapter may include the following sections. (Students are required to perform the testing both manually and automatedly).

8. 1. **Manual Testing**

This is the sample text

6. 1. 1. **System testing**

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended. This is also to check that the system meets the requirements stated earlier. Besides that, system testing will help in finding the errors that may be hidden from the user. There are few types of testing which includes the unit testing, functional testing and integration testing. The testing must be completed before it is being deploy for user to use.

* + 1. **Unit Testing**

Once the system has been successfully developed

* **Unit Testing 1:** Login as FYP Committee as shown in Table 5.1

**Testing Objective:** To ensure the login form is working correctly

*Table 5.1: Login Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify user login after click on the ‘Login’ button on login form with correct input data | Username:  L001  Password:  1234 | Successfully log into the main page of the system as FYP Committee member. | Pass |
| 2. |  |  |  |  |

* **Unit Testing 2:** Edit Profile

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

*Table 5.2: Edit Profile Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. |  |  |  |  |

* + 1. **Functional Testing**

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the module is tested. This is to ensure that the system produced meets the specifications and requirements.

* **Functional Testing 1:** Login with different roles as shown in Table 5.3

**Objective**: To ensure that the correct page with the correct navigation bar is loaded.

*Table 5.3: Login Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as a ‘FYP Committee’ member. | Username: L001  Password: 1234 | Main page for the FYP Committee member is loaded with the FYP Committee navigation bar | Pass |
| 2. |  |  |  |  |

* + 1. **Integration Testing**

Table 5.4 shows the integration testing

*Table 5.4: Integration Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as “FYP Committee” member | Username: L001  Password: 1234 | Login successful and the FYP Committee page with its navigation bar is loaded and in the view profile page | Pass |
| 2. | Upload student record for Project 1 | - | File successfully uploaded and return to the upload page. Student records are updated. | Pass |
| 3. | View supervising student | - | The list of supervisees shown on the screen. | Pass |

* 1. **Automated Testing:**

This is the sample text

* + 1. **Tools used:**

Table 5.5 shows the

*Table 5.5: Tools used*

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool Name** | **Tool Description** | **Applied on [list of related test cases / FR / NFR]** | **Results** |
|  |  |  |  |
|  |  |  |  |

1. **Conclusion and Future Work**

This chapter concludes the project and highlights future work.

1. 1. **Conclusion**
   2. **Future Work**
2. **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. Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. **(Journal paper example)**
  2. Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 **(paper on web)** http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf
  3. Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. **(book reference example)**
  4. Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed**. (web site)**