**SOFTWARE REQUIREMENTS SPECIFICATION**

*For*

**CHAT APPLICATION WITH MESSAGE STORAGE**

Prepared by:-

ATHISH M

HARINI S S

KEERTHANA P

**1. Introduction**

**1.1 Purpose**

The main objective of this document is to illustrate the requirements of the project chat application with stored messages. The document gives the detailed description of the both functional and non-functional requirements.This project describes the hardware and software interface requirements using ER diagrams.

**1.2 Document Conventions**

* Entire document should be justified
* Convention for Main Title
* Font face: Times New Roman
* Font style: Bold
* Font size: 14
* Convention for Sub title
* Font face: Times New Roman
* Font style: Bold
* Font size: 12
* Convention for body
* Font face: Times New Roman
* Font Size: 12

**1.3 Scope of Development Project**

The Chat Application with Stored Messages aims to revolutionize online communication by providing users with a robust platform for real-time messaging and comprehensive message storage. This project facilitates seamless interaction between individuals and groups, ensuring a persistent chat history for enhanced user experience.

This application is tailored for users seeking an efficient and feature-rich messaging solution. It enables users to engage in instant messaging, participate in group conversations, and securely store all communication for future reference.

**1.4 Definitions, Acronyms and Abbreviations**

JAVA -> platform independence

SQL-> Structured query Language

ER-> Entity Relationship

IDE-> Integrated Development Environment

SRS-> Software Requirement Specification

**1.5 References**

**Books:**

**“ Designing Data-Intensive Applications “ by Martin Kleppmann**

**"Java Message Service" by Mark Richards:**

**Websites :**

<https://www.getapp.com/collaboration-software/instant-messaging-chat/f/document-storage/>

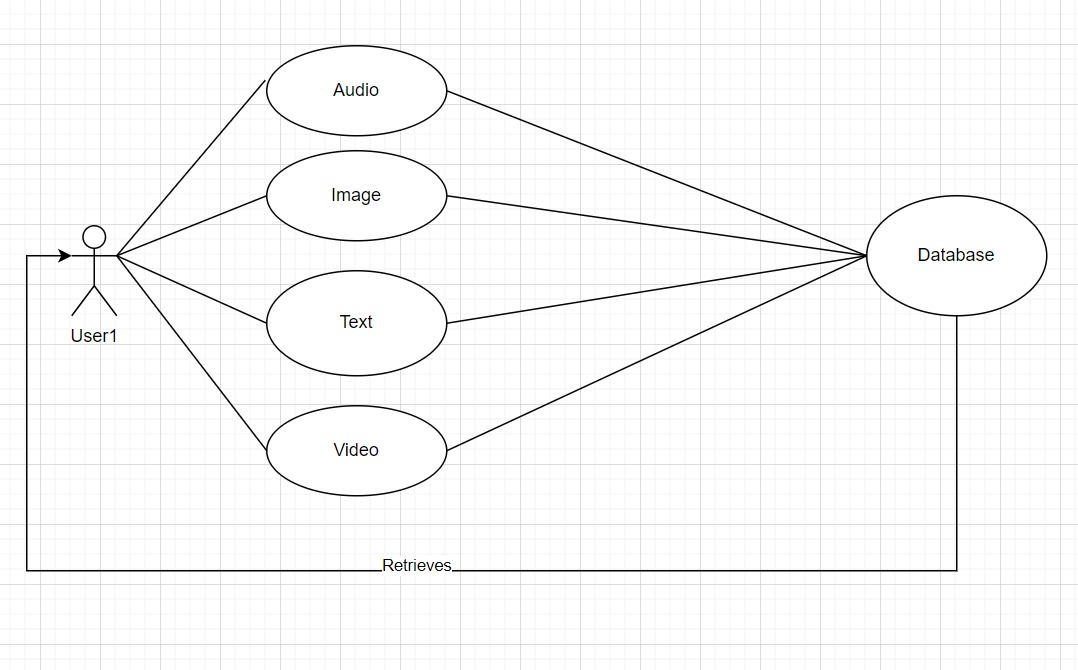
<https://getstream.io/blog/build-chat-messaging-app/>

<https://www.rst.software/blog/how-to-build-a-custom-instant-messaging-app-the-ultimate-guide>

**2. Overall Descriptions**

**2.1 Product Perspective**

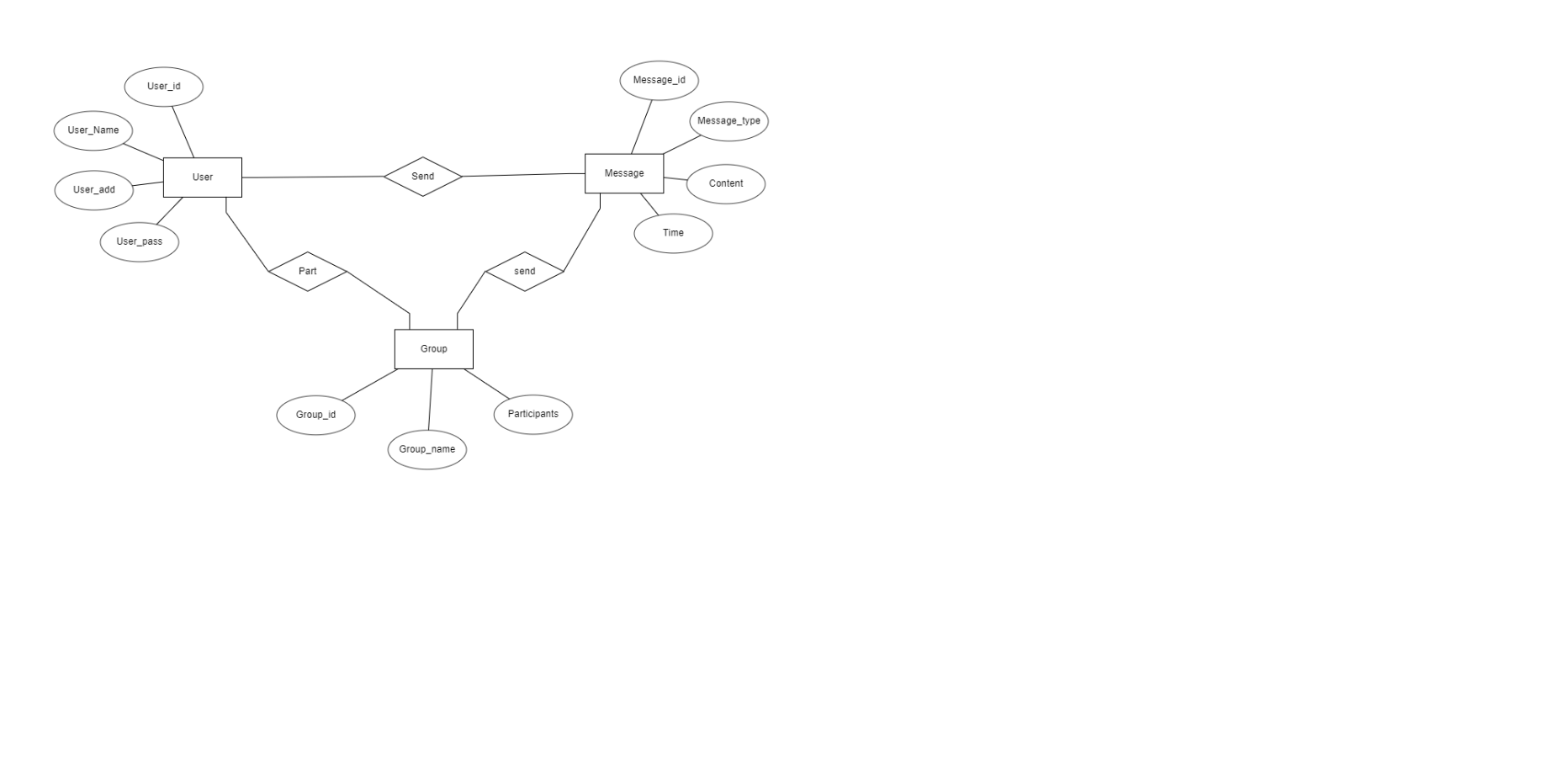
Use case diagram for chat application with stored messages

****

This case diagram provides a high-level overview of the key functionalities and interactions within the chat application. It helps stakeholders understand the different features and how users and administrators interact with the system. Additionally, it can serve as a foundation for more detailed use case descriptions and system design.

**2.2 Product Function**

Entity Relationship Diagram of chat application with stored messages



An Entity-Relationship (ER) diagram is a visual representation of the data model that describes the structure of a database, including the entities, relationships between entities, and attributes of entities. For a chat application with stored messages, the ER diagram would involve entities such as User, Message, and possibly others depending on the features of the application.

**2.3 User Classes and Characteristics:**

In a chat application with stored messages, you can identify different user classes based on their roles and characteristics.

**Regular User:**

* **Characteristics:**
  + Can send and receive messages.
  + Has a unique username or identifier.
  + Can create and participate in one-on-one and group chats.
  + Can view and manage their chat history.
  + May have a profile with personal information.
* **Administrator:**
  + **Characteristics:**
    - All characteristics of a regular user.
    - Has additional privileges for managing the overall system.
    - Can moderate user activities.
    - Can create, edit, or delete chat groups.
    - Has access to system logs and analytics.

**2.4 Operating Environment**

The term "Operating Environment" generally refers to the conditions and surroundings in which a software application operates. For a chat application with stored messages, the operating environment includes the hardware, software, and network infrastructure necessary for the application to function.

**2.5 Assumptions and Dependencies**

**Assumptions:**

1. **User Authentication:**
   * Assumption: Users will provide accurate and valid information during the registration/authentication process.
   * Implication: User identity and access control are dependent on accurate user registration.
2. **Network Connectivity:**
   * Assumption: Users have reliable internet access.
   * Implication: The application relies on internet connectivity for real-time messaging and data synchronization.
3. **Device Compatibility:**
   * Assumption: Users are using compatible devices (smartphones, tablets, computers) with supported operating systems and browsers.
   * Implication: The application's user interface and features are designed for specific device types and operating systems.
4. **Message Delivery:**
   * Assumption: Messages will be successfully delivered and received by the intended recipients.
   * Implication: The application assumes a reliable messaging infrastructure and may not handle cases of message loss due to network issues.
5. **Data Privacy and Security:**
   * Assumption: Users will not share their login credentials, and the application is secure against unauthorized access.
   * Implication: Data privacy and security measures depend on user compliance with recommended security practices.

**Dependencies:**

1. **Database System:**
   * Dependency: The application relies on a specific database system (e.g., MySQL, MongoDB) for storing user data, messages, and other relevant information.
2. **Server Infrastructure:**
   * Dependency: The application is dependent on server infrastructure, whether hosted on-premises or in the cloud (e.g., AWS, Azure).
3. **Authentication Service:**
   * Dependency: The application depends on an authentication service for user registration, login, and session management.
4. **Network Services:**
   * Dependency: The application relies on network services and protocols (e.g., WebSocket) for real-time communication.
5. **Third-Party APIs:**
   * Dependency: If the application uses third-party services (e.g., push notification services, mapping APIs), it depends on the availability and proper functioning of these services.
6. **Development Frameworks and Libraries:**
   * Dependency: The application is built using specific programming languages, frameworks, and libraries (e.g., Node.js, React, Django).
7. **Operating System:**
   * Dependency: The application is dependent on the underlying operating system(s) where it is deployed (e.g., Linux, Windows).
8. **Browser Compatibility:**
   * Dependency: Web-based applications depend on the compatibility of browsers (e.g., Chrome, Firefox) with the application's frontend code.
9. **User Compliance:**
   * Dependency: The application's security and privacy features depend on users following recommended security practices.
10. **Regulatory Compliance:**
    * Dependency: The application must comply with relevant legal and regulatory requirements related to data privacy and security.

**2.6 Requirement**

**System Requirements:**

1. **Scalability:**
   * Design the system to scale with increasing user and message loads.
2. **Reliability:**
   * Ensure high system availability and reliability to prevent service interruptions.
3. **Performance:**
   * Optimize system performance for quick message delivery and retrieval.
4. **Cross-Platform Compatibility:**
   * Support multiple platforms, including web browsers, mobile devices (iOS, Android), and desktop applications.

**Security Requirements:**

1. **End-to-End Encryption:**
   * Implement end-to-end encryption for secure message transmission.
2. **User Data Protection:**
   * Safeguard user data and ensure compliance with data protection regulations.
3. **Authentication Security:**
   * Employ secure authentication methods and protect against unauthorized access.
4. **Secure File Transfer:**
   * Ensure the secure transfer of multimedia and file attachments.

**5.4 Requirement attributes**

There may be multiple admins creating the project, all of them will have the right to create changes to the system. But the members or other users cannot do changes . The project should be open source .The Quality of the database is maintained in such a way so that it can be very user friendly to all the users of the database. The user be able to easily download and install the system

**5.5 Business Rules**

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data.This includes the rules and regulations that the System users should abide by. This includes the cost of the project and the discount offers provided. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations.

**5.6 User Requirement**

The users of the system are members and Librarian of the university who act as administrator to maintain the system. The members are assumed to have basic knowledge of the computers and internet browsing. The administrators of the system should have more knowledge of the internals of the system and is able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, user manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems. The admin provides certain facilities to the users in the form of:- ¬ Backup and Recovery ¬ Forgot Password ¬ Data migration i.e. whenever user registers for the first time then the data is stored in the server ¬ Data replication i.e. if the data is lost in one branch, it is still stored with the server ¬ Auto Recovery i.e. frequently auto saving the information ¬ Maintaining files i.e. File Organization ¬ The server must be maintained regularly and it has to be updated from time to time

**6. Other Requirements**

**6.1 Data and Category Requirement**

There are different categories of users namely teaching staff, Librarian, Admin, students etc. Depending upon the category of user the access rights are decided.It means if the user is an administrator then he can be able to modify the data,delete, append etc. All other users except the Librarian only have the rights to retrieve the information about database. Similarly there will be different categories of books available. According to the categories of books their relevant data should be displayed. The categories and the data related to each category should be coded in the particular format.

**6.2 Appendix**

A: Admin, Abbreviation, Acronym, Assumptions; B: Books, Business rules; C: Class, Client, Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; L: Library, Librarian; M: Member; N: Non-functional Requirement; O: Operating environment; P: Performance,Perspective,Purpose; R: Requirement, Requirement attributes; S: Safety, Scope, Security, System features; U: User, User class and characteristics, User requirement.

**6.3 Glossary**

The following are the list of conventions and acronyms used in this document and the project as well: ¬ Administrator: A login id representing a user with user administration privileges to the software ¬ User: A general login id assigned to most users ¬ Client: Intended users for the software ¬ SQL: Structured Query Language; used to retrieve information from a database ¬ SQL Server: A server used to store data in an organized format ¬ Layer: Represents a section of the project ¬ User Interface Layer: The section of the assignment referring to what the user interacts with directly ¬ Application Logic Layer: The section of the assignment referring to the Web Server. This is where all computations are completed ¬ Data Storage Layer: The section of the assignment referring to where all data is recorded ¬ Use Case: A broad level diagram of the project showing a basic overview ¬ Class diagram: It is a type of static structure diagram that describes the structure of a system by showing the system’s cases, their attributes, and the relationships between the classes ¬ Interface: Something used to communicate across different mediums ¬ Unique Key: Used to differentiate entries in a database

**6.4 Class Diagram**

A class is an abstract, user-defined description of a type of data. It identifies the attributes of the data and the operations that can be performed on instances (i.e. objects) of the data. A class of data has a name, a set of attributes that describes its characteristics, and a set of operations that can be performed on the objects of that class. The classes’ structure and their relationships to each other frozen in time represent the static model. In this project there are certain main classes which are related to other classes required for their working. There are different kinds of relationships between the classes as shown in the diagram like normal association, aggregation, and generalization.

