**DBMS Course Project Synopsis**

**Project Members :**

1. **Vasu Sharma - 111803127**
2. **Vasvi Gupta - 111803128**

**Problem Statement :** We aim to develop a **Real Time Chat Application** like WhatsApp or Telegram.

**Introduction:** This can be split up into 2 parts as follows :

**Idea Build Up :** The problem with current chat applications is that they store nothing in their databases. All the chats and media files are stored in server RAM for limited time and then ultimately stored only on the user's phone. So, this can be problematic at times, as the files can be deleted by users which are required later.

**Proposed Solution:** So, we are aiming to make a Chat Application which will use databases heavily and provide a reliable solution for resource management and information retrieval. This will store all important messages and media in the database.This will really come handy especially in official communication where information retrieval and accessibility matters a lot.

**Functional Requirements Of The System**

The Project is aimed to achieve the following features :

1. **Real Time Chat - One to One:** Users will be able to chat with other people on the same app and the messages will be stored on the server so that they can fetch the messages on different devices also.

Note that users will be able to fetch messages without being active on other devices, as opposed to WhatsApp web presently which presently fetches everything from Mobile

1. **Media and File Sharing :** Media and file sharing like photos, documents will be possible for users using the app and will be preserved on the database for future downloading and accessibility.
2. **Group Creation and Chatting :** We will also provide group creation and sending messages in bulk to a large number of people.
3. **Push Notifications :** Everytime a message arrives, the users will be notified.
4. **Github Integration :** This is a future perspective where users will be able to add their github projects’ repositories to a group and any changes to that repository will eventually appear as a message from github on groups.

**Tech Stack Proposed:**

Database : MYSQL

FrontEnd : React , PWA

Backend : Express Server(Node JS)

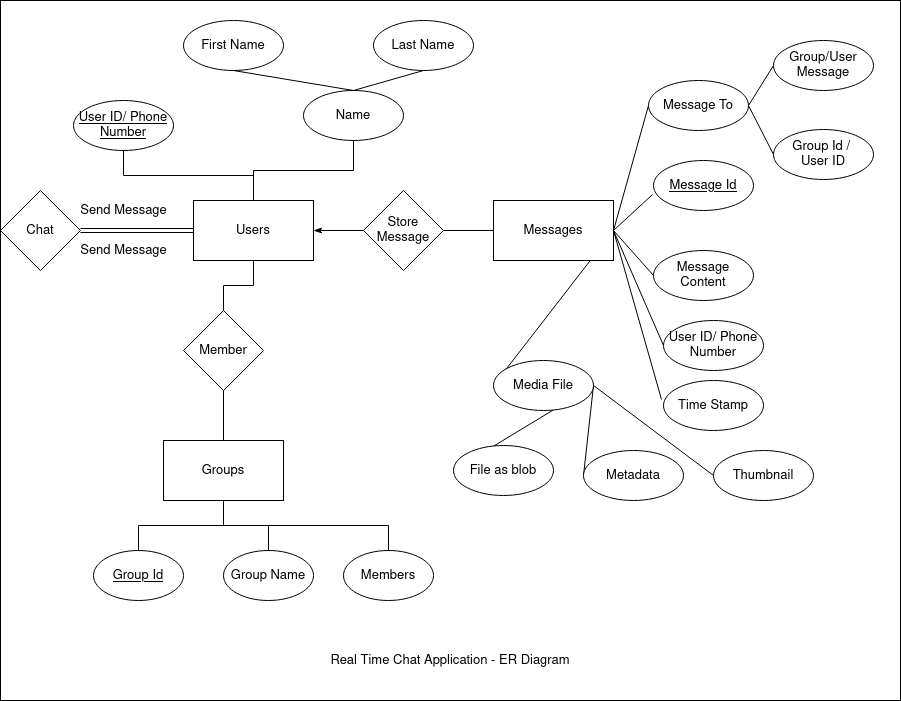
(In Node JS , the mysql npm package handles the interaction with the database.)

Hosting Services:Amazon AWS

Platforms : Web,Android

For cloud Database : Google Cloud SQL / Amazon RDS/ MySQL on server

**ER Diagram**

****

**Entities:**

1. **Users =** *(User ID/Phone Number, Name)*

* **Primary Key :** *User ID/Phone Number*
* **One-to-many relationship with** *Messages* **via the relation** *Store Message* **as one user can store several messages at a time**
* **Many-to-many relationship with** *Groups* **via the relation** *Member* **as a user can be a member of several groups and vice-versa**
* **Participation of** *Users* **in** *Chat* **is total**
* **Role of how** *Users* **entities interacts with the** *Chat* **relationship set : Send message**
* *Name* **: Composite attribute with component attributes** *First Name* **and** *Last Name*

1. **Groups =** *(Group ID, Group Name, Members)*

* **Primary Key :** *Group ID*
* **Many-to-many relationship with** *Users* **via the relation** *Member***, as a group can contain various users and vice-versa**

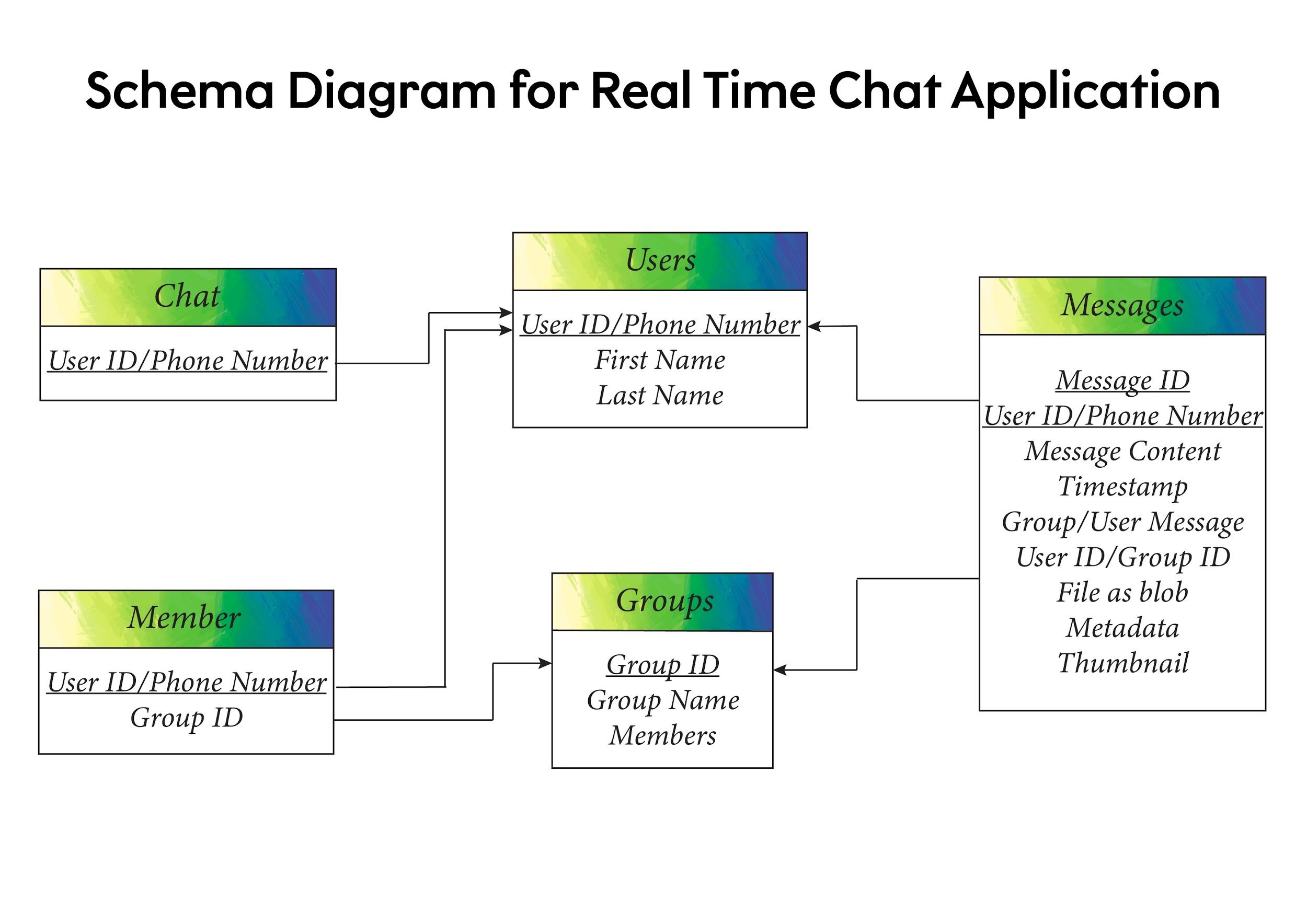
1. **Messages =** *(Message ID, Message To, Message Content, Time Stamp, Media File)*

* **Primary Key :** *Message ID*
* **Many-to-one relationship with Users via the relation** *Store Message* **as a message is associated with at most one user at a time**
* *Message To* **: Composite attribute with attributes** *Group/User Message* **and** *User/Group ID*
* *Media File* **: Composite attribute with attributes** *File as blob, Metadata* **and** *Thumbnail*

**Relationship Sets:**

1. **Chat :** This relation specifies how users interact(chat) with each other. User1 sends a message to User2 who in turn replies to User1 via the *Chat* relation.
2. **Member :** This relation specifies the members of a group or in other words, which groups a user is a part of.
3. **Store Message :** This relation specifies the details of the messages which are associated with a particular user.

**Relational Schema**

****

**Relations:**

1. **Users =** *(User ID/Phone Number, First Name, Last Name)*
2. **Groups =** *(Group ID,Group Name,Members)*
3. **Messages =** *(Message ID,User ID/Phone Number,Message Content,Timestamp,Group/User Message,User ID/Group ID,File as blob,Metadata,Thumbnail)*
4. **Chat =** *(User ID/Phone Number)*
5. **Member =** *(User ID/Phone Number,Group ID)*

**Functional dependencies:**

1. **Users :**

{*User ID/Phone Number*} -> {First Name, Last Name}

1. **Groups:**

{*Group ID*} -> {Group Name}

1. **Messages:**
   1. {*Message ID}* -> {Message Content,File as blob,Thumbnail,Metadata}
   2. *{Message ID,User ID/Phone Number}* -> {Timestamp,Group/User Message,User ID/Group ID}
2. **Chat:**

No functional dependencies

1. **Member:**

No functional dependencies

**Normalized Relations:**

1. **Users =** *(User ID/Phone Number, First Name, Last Name)*
2. **Groups =** *(Group ID,Group Name,Members)*
3. **Messages =** *(Message ID,User ID/Phone Number,Timestamp,Group/User Message,User ID/Group ID)*
4. ***MessageContent =*** *(Message ID,Message Content,File as blob,Thumbnail,Metadata)*
5. **Chat =** *(User ID/Phone Number)*
6. **Member =** *(User ID/Phone Number,Group ID)*

**Steps for Normalisation of Relations:**

**1NF -** All the relations were already in 1NF form as all the attributes were **atomic** and **single valued**. (Single valued attribute constraint will be taken care of during the development process).

**2NF**

* Relations **Users, Groups, Chat** and **Member** were already in 2 NF.
* In **Messages** Relation, as *Message Content, File as Blob ,Metadata and Thumbnail* attributes were functionally dependent on *Message ID* , which is a subset of Candidate Key (*Message ID,User ID/Phone Number),* so the relation is not in 2NF. So the relation is decomposed into a new relation **MessageContent** with attributes *(Message ID,Message Content,File as blob,Thumbnail,Metadata)*

**3NF -** All the relations are in 3NF as there is no functional dependency from non- prime to other attributes.