Questions

1. Create a Schema Design to store WhatsApp Group Messages. (Low-level design of WhatsApp group messages). Feel free to base this schema on any SQL/NoSQL database.

2. Write a simple server-side application using NodeJS (You may use Javascript or Typescript as language). This application should run a simple HTTP server on port 3000 and should contain the following API -

a. API to load all the group messages in a paginated manner.

b. API to create a message in the group (post a message in the group chat)

Solution: Several elements and technologies would be used in the low-level design of a WhatsApp group chat functionality, including:

Database: Data about the groups, group members, and messages would be kept in a database like MySQL or MongoDB. The database's schema would comprise tables for messages, groups, and group members as well as a table that would map messages to groups via a foreign key relationship.

Server: Incoming requests are handled by a server-side program, which also manages database operations like message creation and retrieval. A framework like Express and a technology like Node.js might be used to create this.

Client-side apps (such as mobile apps or web apps) can communicate with the server and the database by using a set of APIs that the server exposes. These APIs would take care of operations like sending and receiving messages, adding and removing group members, and querying a user's membership in various groups.

WebSocket or Push notification: The application would use WebSockets or Push notifications to notify users when a new message is added to a group in order to deliver real-time updates to the user.

Security: To ensure that only authorized users can access the groups and messages, the application would use authentication and authorization techniques.

Storage: The program would use cloud storage services like AWS S3 or Google Cloud Storage to store the media files, such as pictures and movies.

A load balancer will be used to manage the traffic and distribute it among several servers.

Handling of edge cases and corner cases in the code

Solution:
Handling a large number of group members: If a group has a large number of members, it can cause issues with performance and scalability. To handle this, the application can use load balancing and sharding techniques to distribute the load among multiple servers. Additionally, the application can also implement pagination and filtering to retrieve only the relevant members from the database.

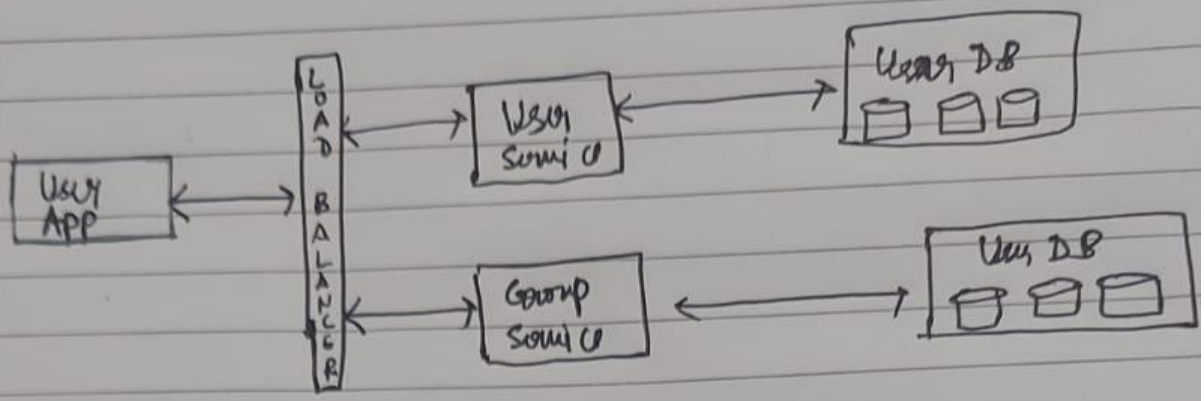
Handling a large number of messages: A group with a large number of messages can cause issues with performance and scalability. To handle this, the application can use pagination and filtering to retrieve only the relevant messages from the database, and also implement archiving mechanism to move old messages to different storage.

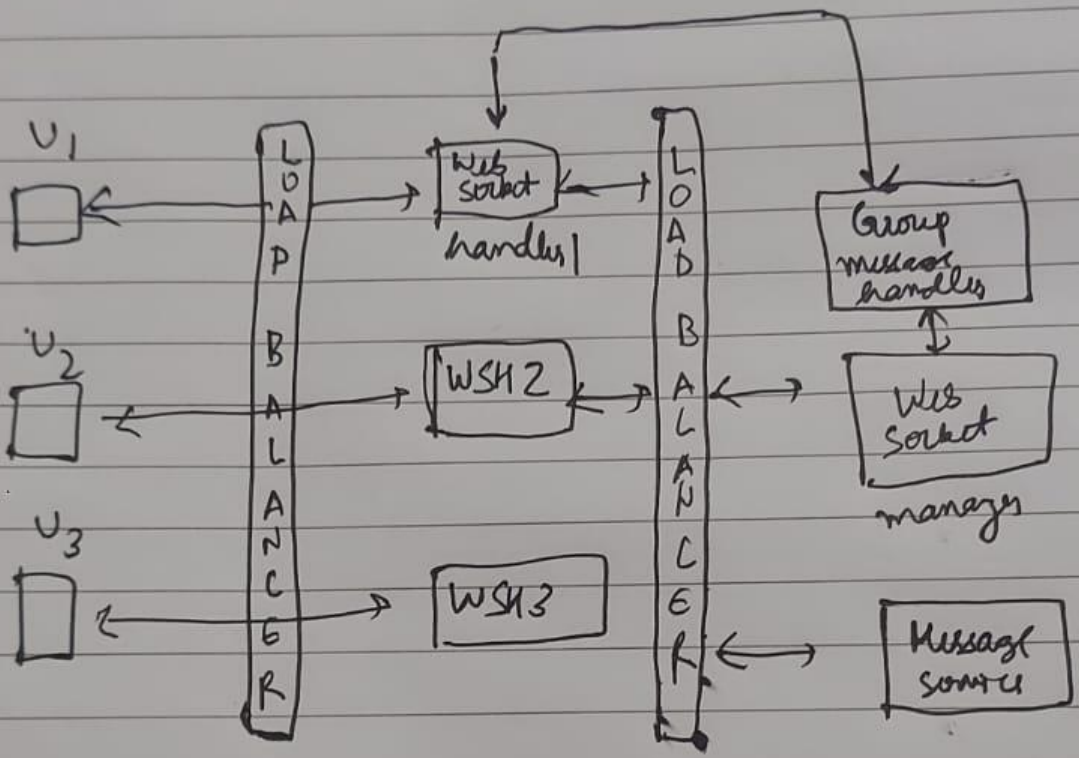
Handling media files: Media files such as images and videos can consume a large amount of storage. To handle this, the application can use cloud storage solutions like AWS S3 or Google Cloud Storage to store the media files. Additionally, the application can also implement compression and optimization techniques to reduce the file size before storing it.

Handling network connectivity: In case of poor network connectivity, the application should be able to handle the case when the message is not delivered in real-time and handle the retries and message queueing mechanism.

Handling security: The application should implement various security measures such as encryption, authentication, and authorization to protect user data and prevent unauthorized access. Additionally, the application should handle cases such as SQL injection and cross-site scripting attacks.

Schema design





WSH :- Websocket handler