Slak Application

Apply a **Two-tier Client-Server** Architecture

Presented by

Razan Aljelify
Ruof Alsheddi
Layan Aldaghfaq

Presented to

Dr.Mohammed assiri

What is Slack?

Slack is a messaging app for business that connects people to the information they need. By bringing people together to work as one unified team, Slack transforms the way organizations communicate.

What is Two-tier Client-Server Architecture?

A **'tier'** commonly refers to the logical or functional separation of software into layers on different physical locations or hardware.

The two-tier architecture is based on the Client-Server model. It consists of a **Client-Application tier** and a **Database tier**.

The Client-Application server → communicates directly with the Database server.

Data or information transfer between the two components is fast due to the absence of middleware.

- ✓ **The Client-Application contains** the codes for interfacing with the user and for saving data in the database server.
- ✓ **The Client-Application sends** the request to the server, and it processes the request and sends it back with data.
- ✓ **The client-Application layer can** be built using languages such as C, C++, Java, Python, PHP, and Rails. NET.

This means the client Application handles both the Presentation layer (application interface) and the Application layer (logical operations).

The Database Server → handles the data management layer.

- ✓ **Data management layer consists** of data storage (database or file system) and methods for storing and retrieving data from the data storage.
- ✓ **Commonly used databases include** MySQL, MongoDB, PostgreSQL, and SQLite. Hosting is possible on-premises or in the cloud.

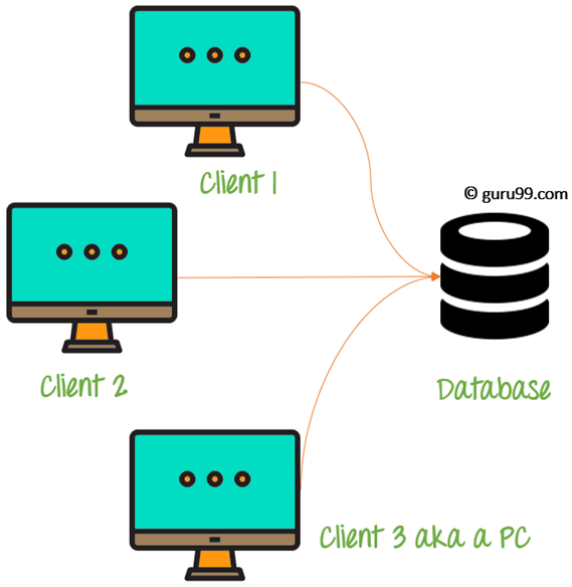


Figure to illustrate

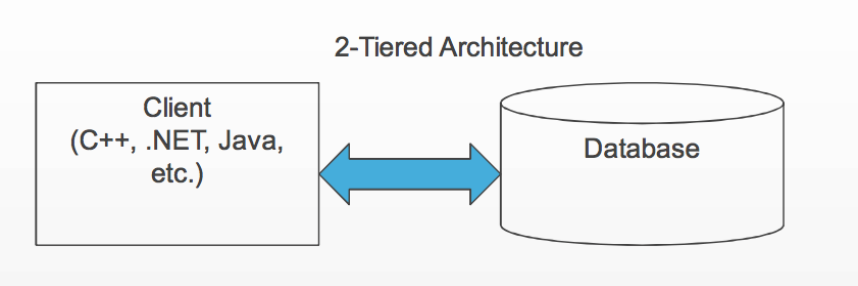


Figure to illustrate

Assumptions that we considered:

- Slack is a Real-Time Messaging API, a WebSocket-based API that allows you to receive events from Slack in real-time and send messages as users.
- Considering more than a billion messages are sent on Slack weekly.
- Slack used a new database table called “shared_channels” as a bridge to connect workspaces in a shared channel.

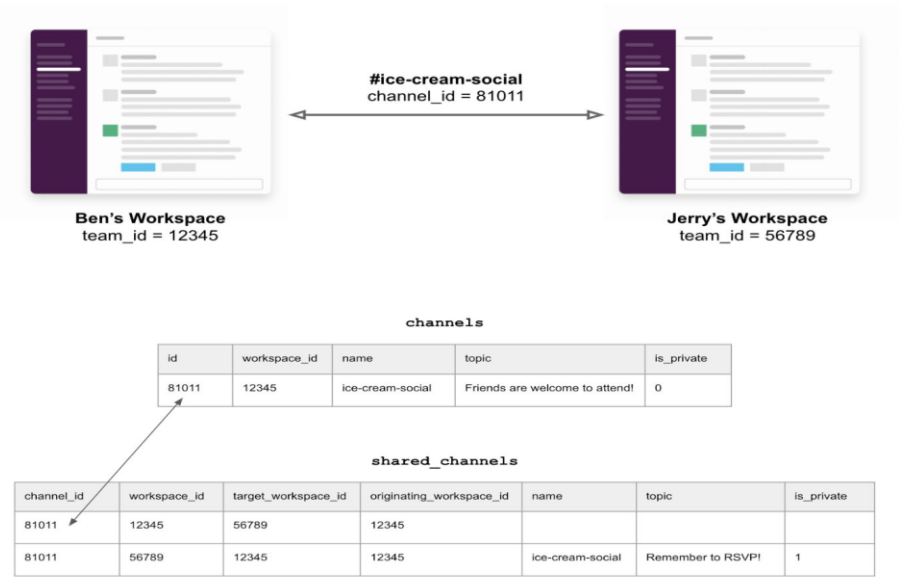


Figure to illustrate

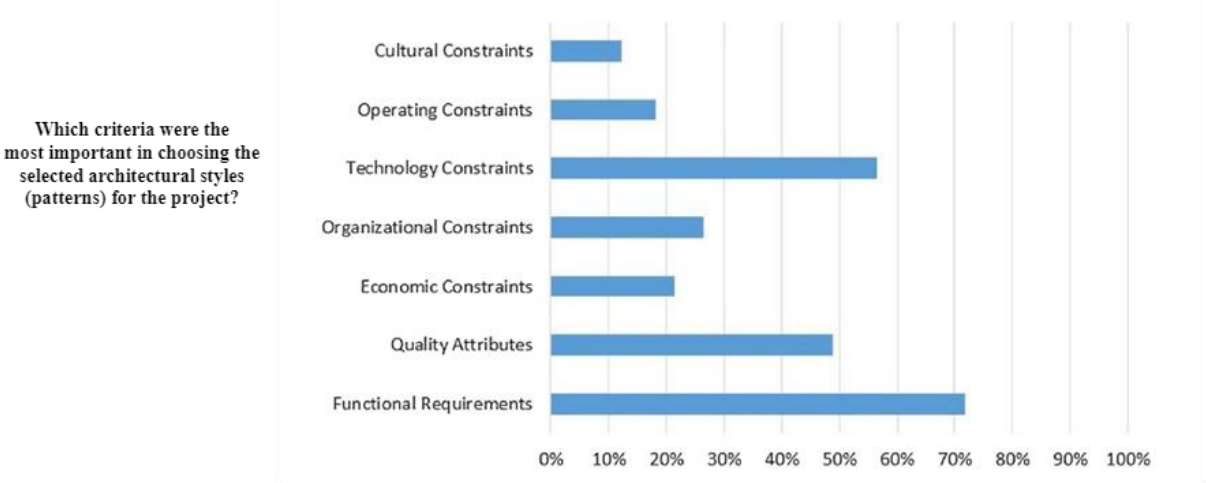
- Privacy challenges.
- Slack is built for large organizations.
- Performance: fast communication, etc.
- Ability to maintain.
- Ease of modification.

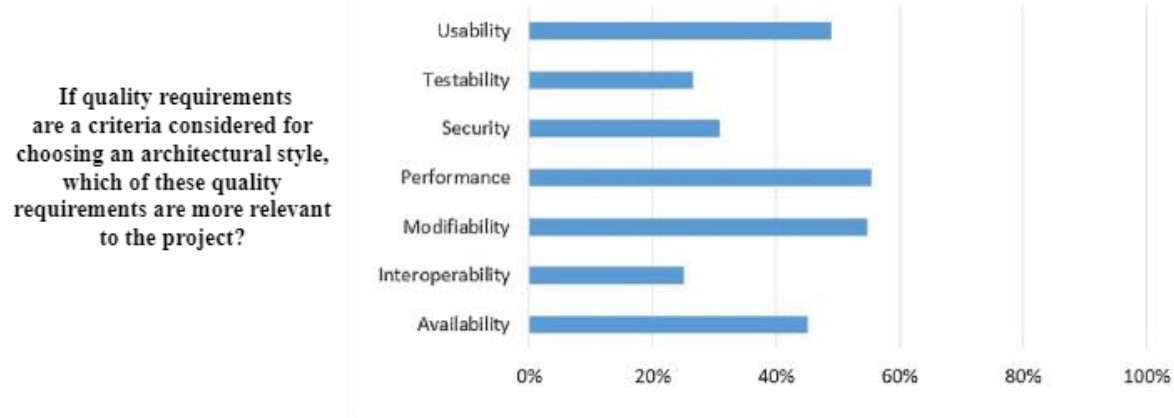
Alternative considered:

1. Client-Server Architecture.
2. Three-Tier Architecture.
3. N-Tier Architecture.
4. The Repository Architecture.

What are the criteria used to select the Two-tier Client-server Architecture among others?

The architect selects patterns **based on their ability to support the majority of the requirements of the system.** Patterns embody the high-level structure of the system. Nevertheless, selected patterns (in their standard structures) may not support the entire requirements list. In order to address the remaining quality requirements that are not prompted by the selected patterns and in order to handle trade-off analysis, architectural tactics can then be incorporated into the higher structure to address particular qualities locally.



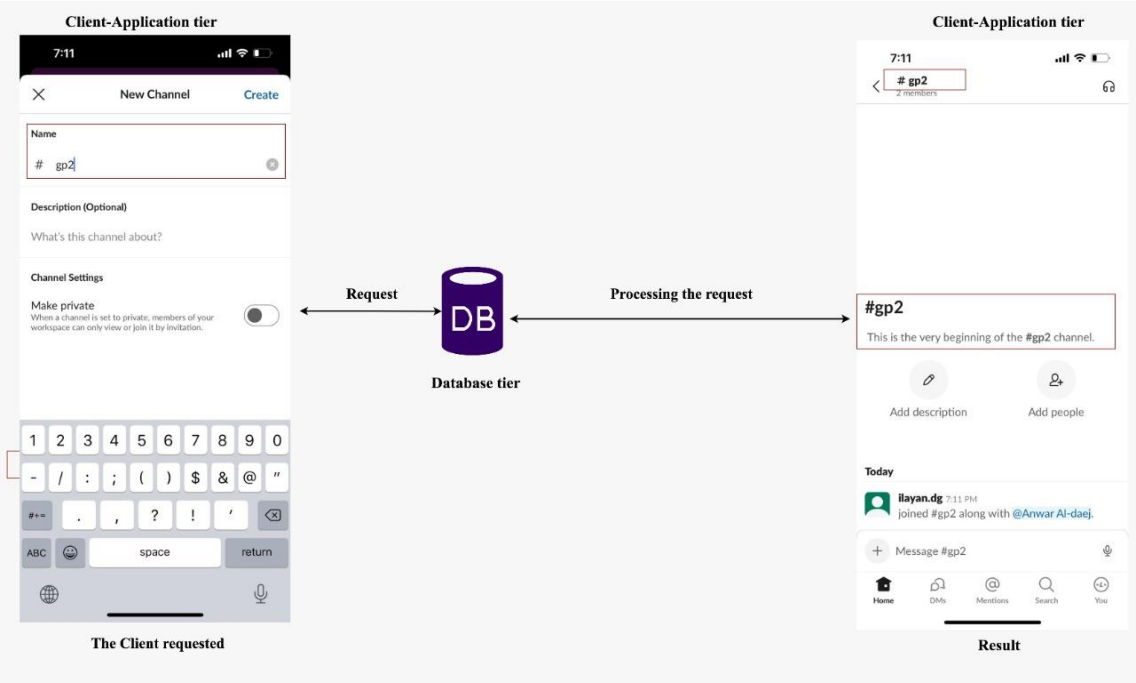


Moreover, cost, number of users, Performance, and the assumption mentioned before had a role in choosing the specific pattern.

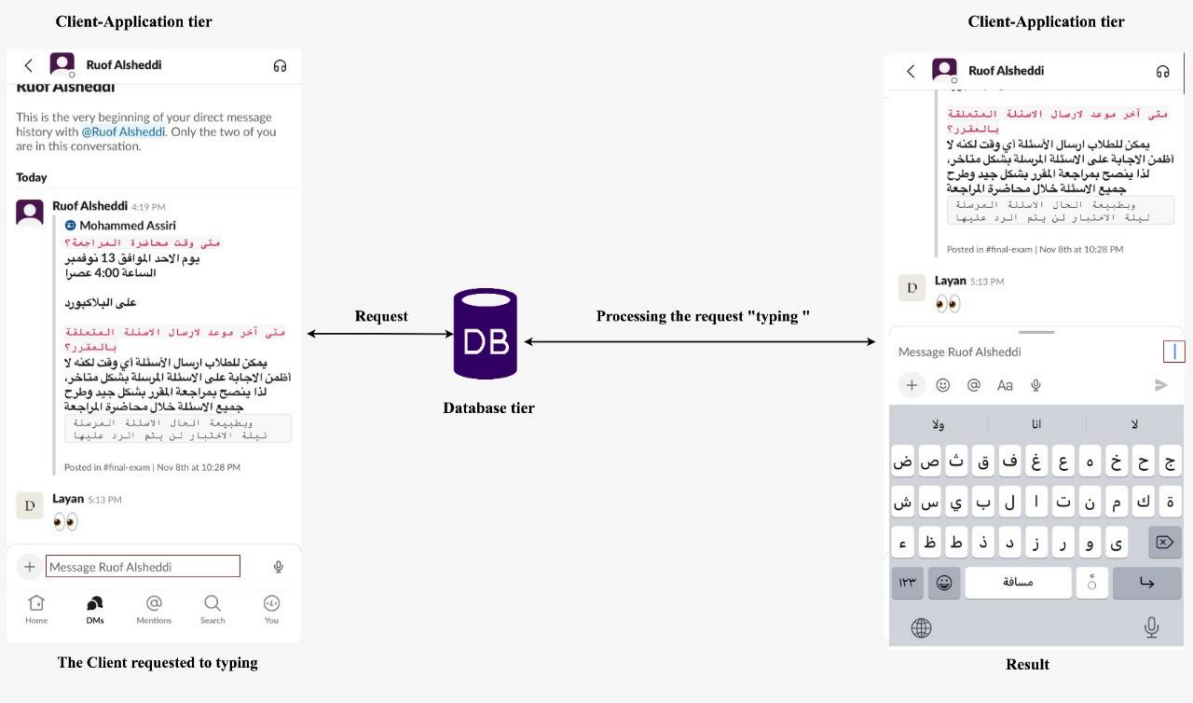
According to the Slack engineering website, Slack implements a client-server architecture, but in this tutorial, we will see HOW to apply **Two-tier Client-Server Architecture.**

In previous functional requirements that we are provided:

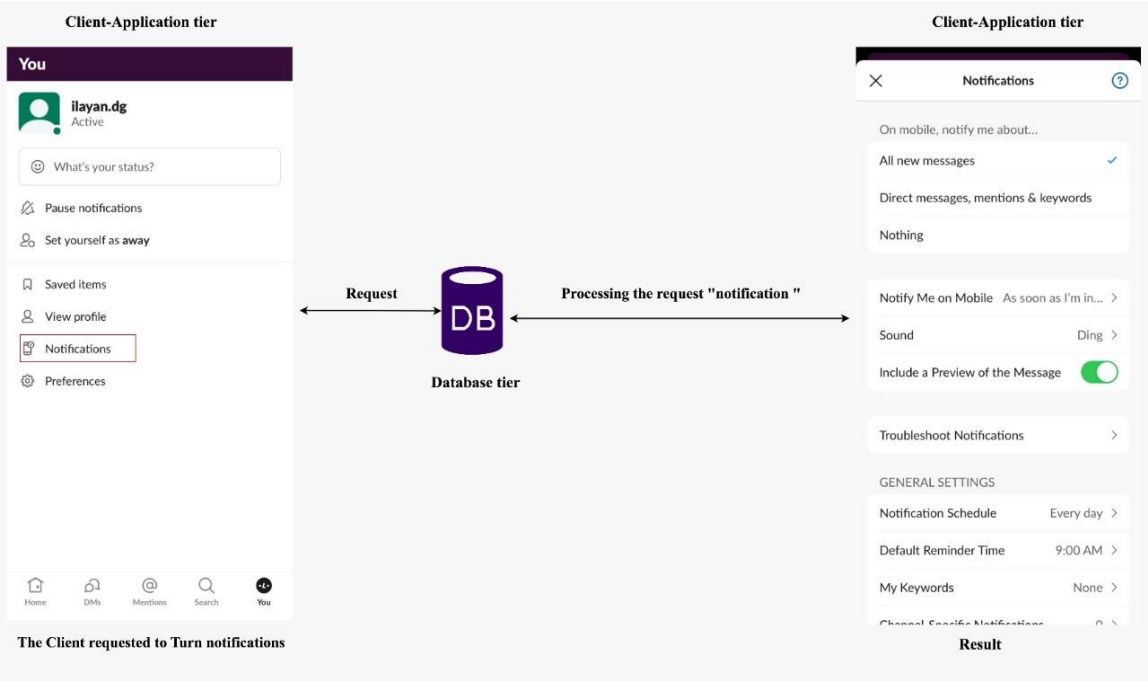
FR 1 : " The system shall support naming channels"



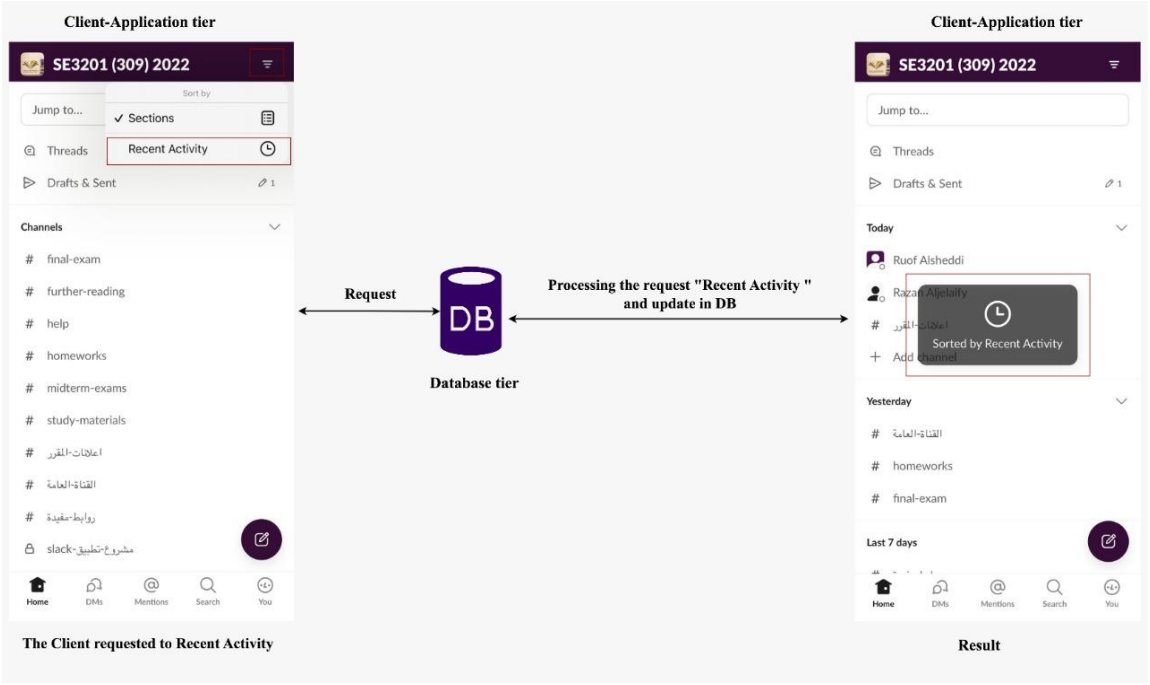
FR 2 : " The system shall show typing indicators when typing"



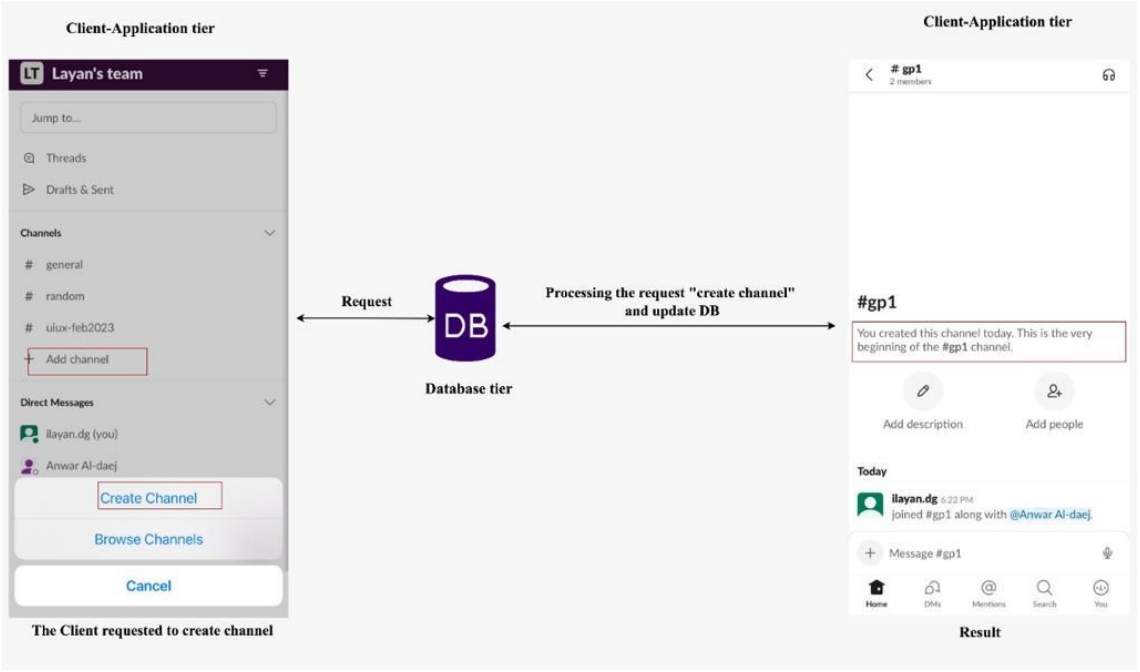
FR 3 :" The system shall send notifications to users when they receive messages"



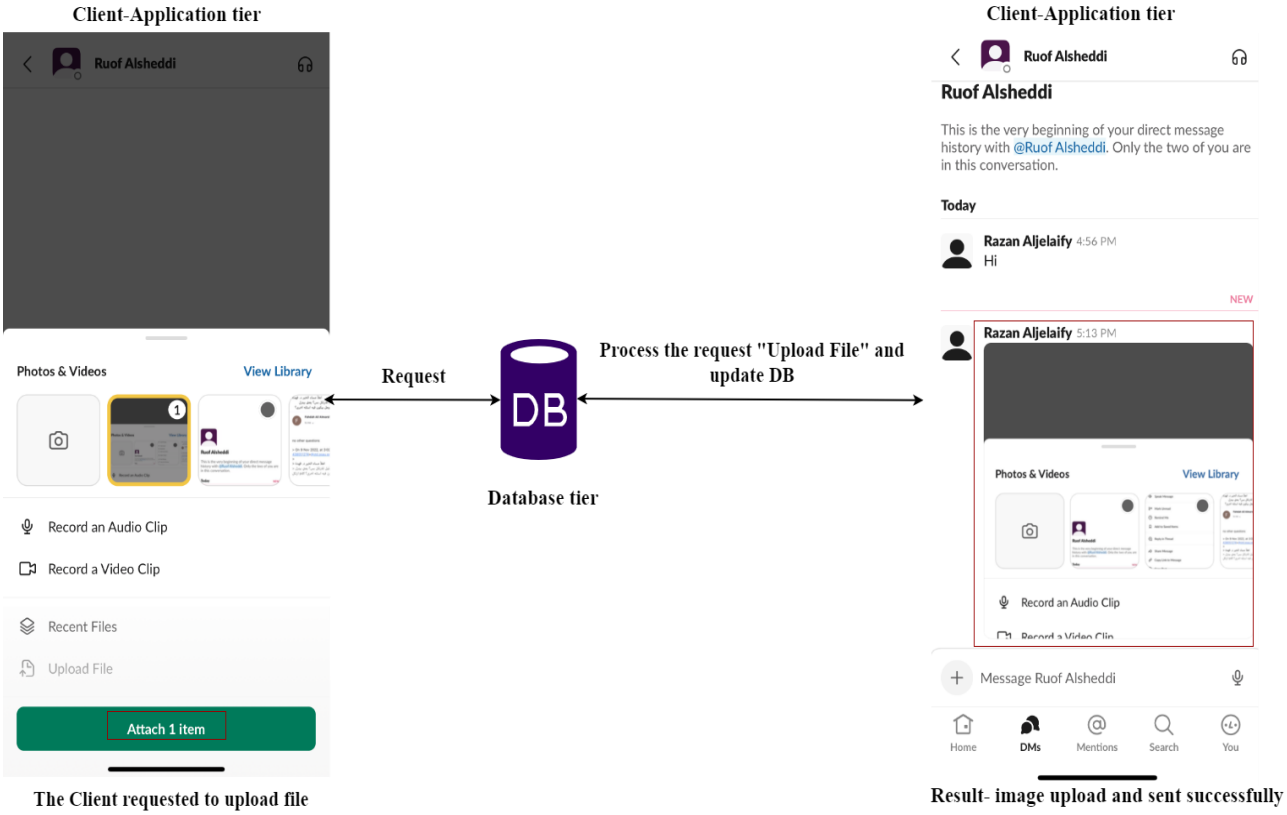
FR 4 :" he system shall arrange messages in chronological order."



FR 5 : " A user should set up their own channels"



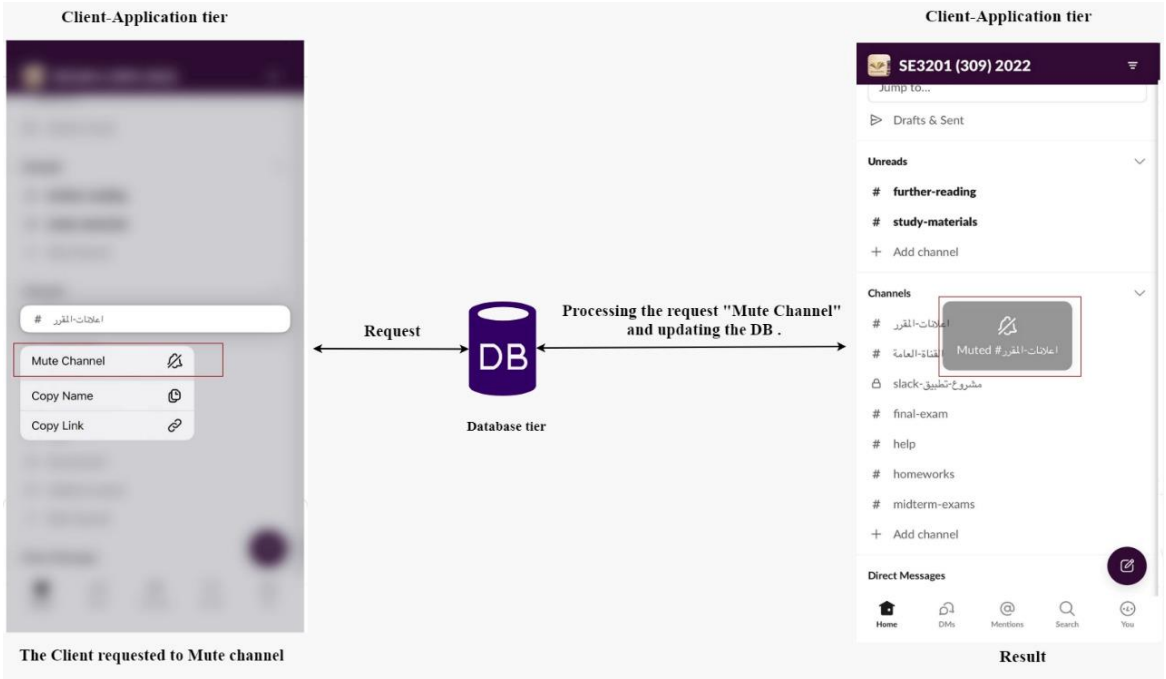
FR 6 :" A user should send files, images, and videos"



The Client requested to upload file

Result- image upload and sent successfully

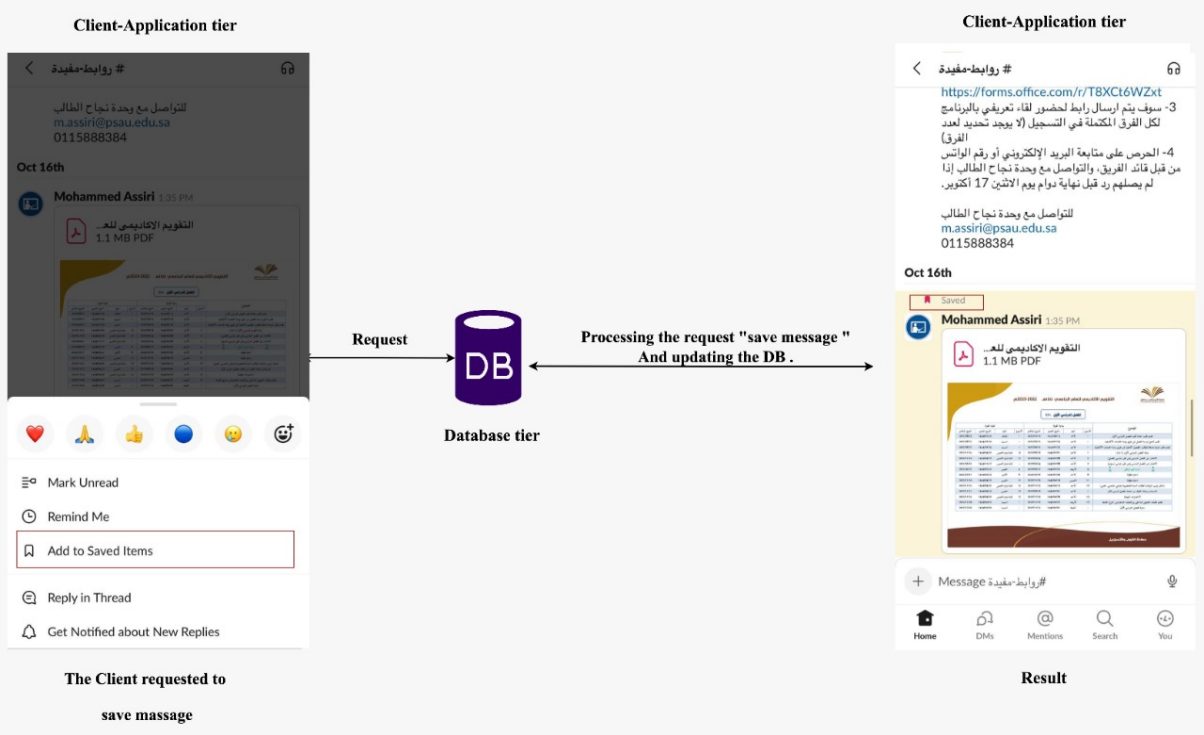
FR 7 : " A user should mute conversion/channel."



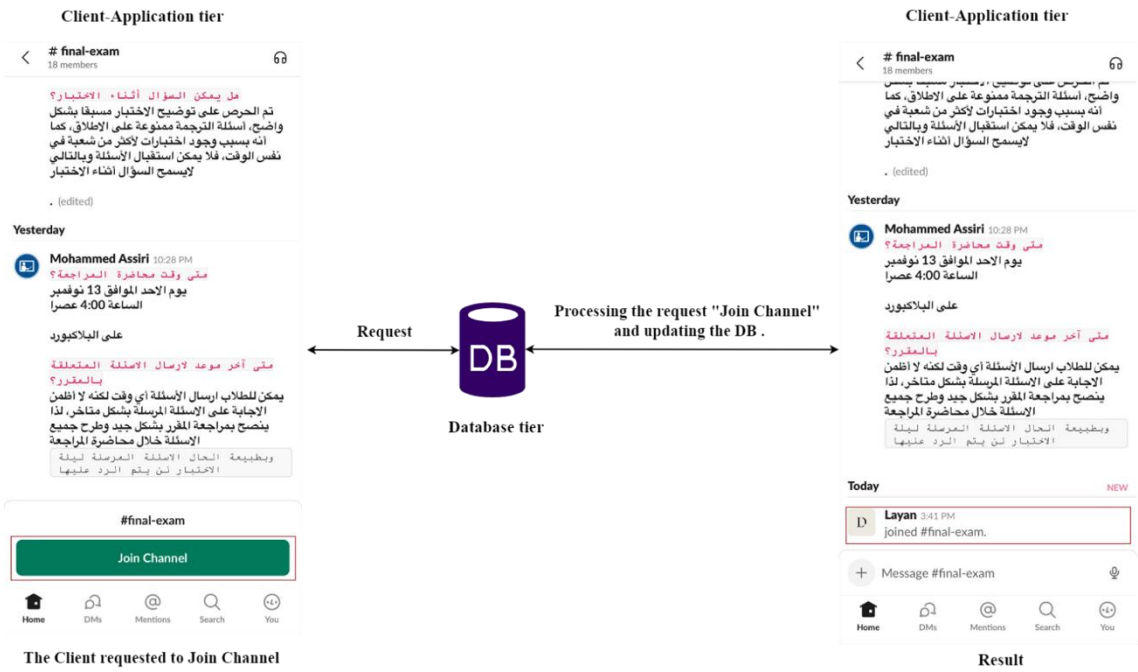
The Client requested to Mute channel

Result

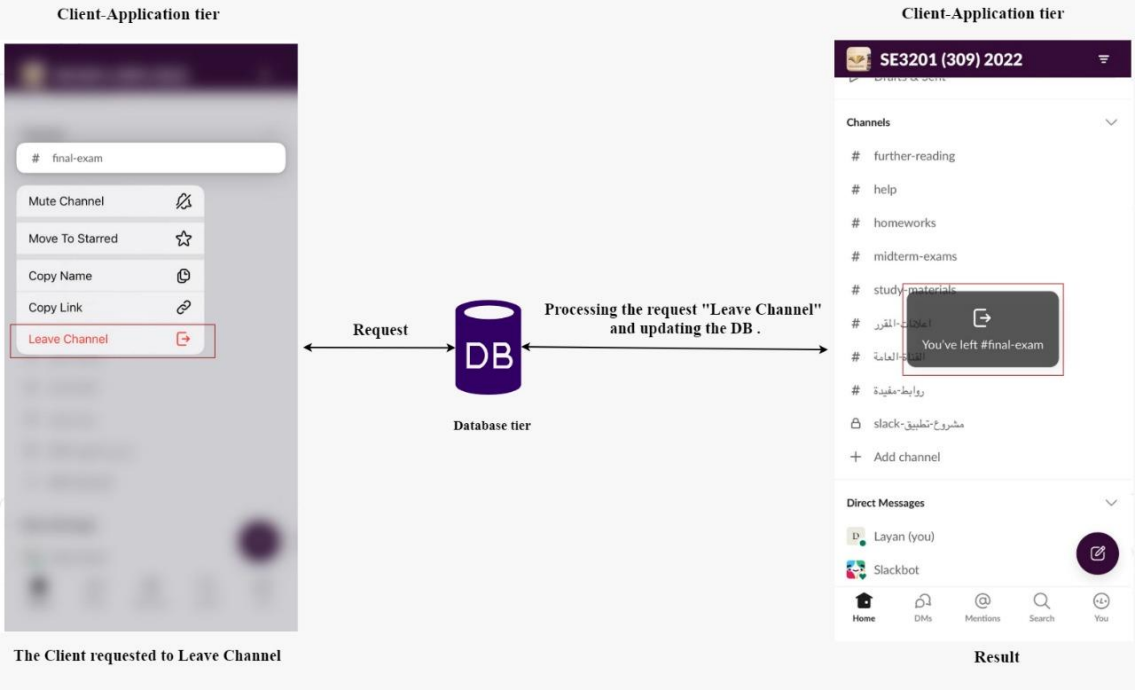
FR 8 : " A user should save the message to saved items"



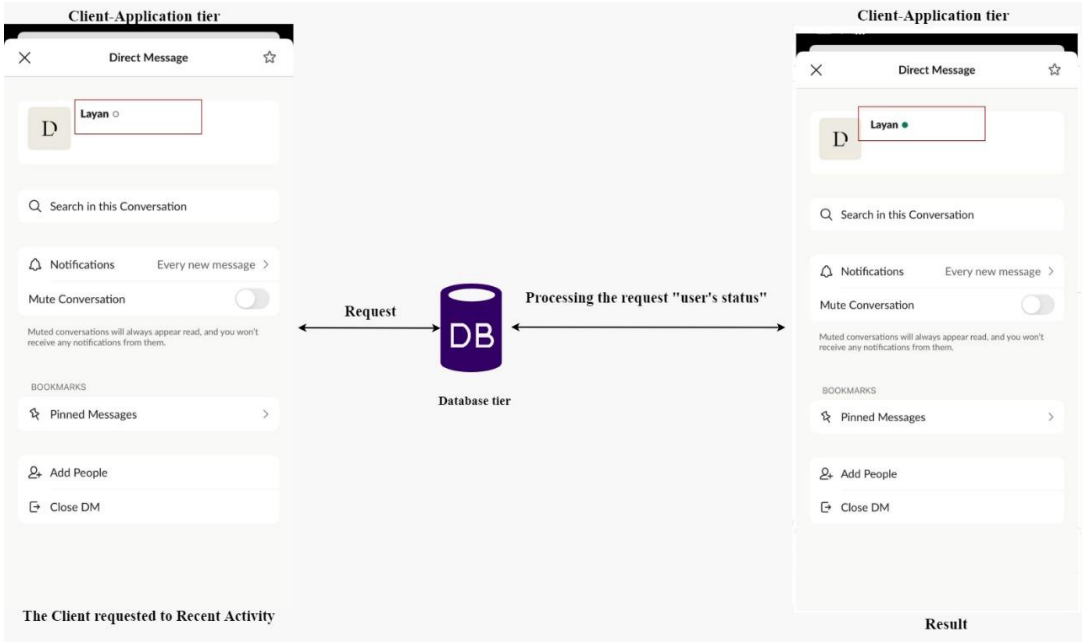
FR 9 :" A user should Join a channel."



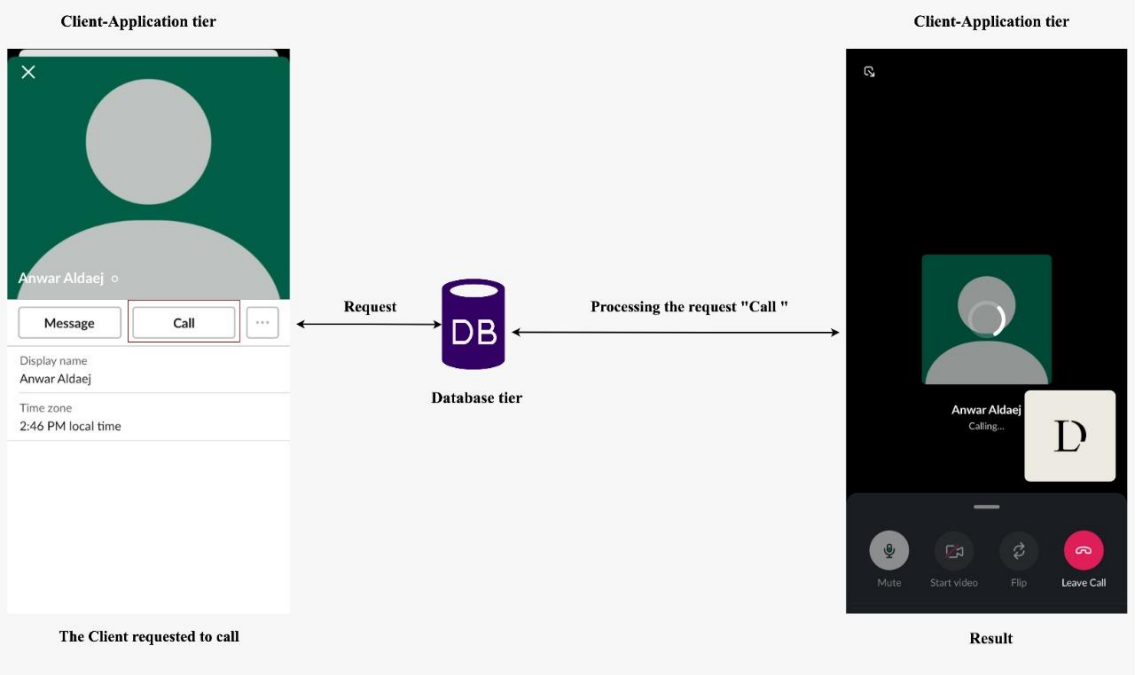
FR 10 :" A user should leave a channel"



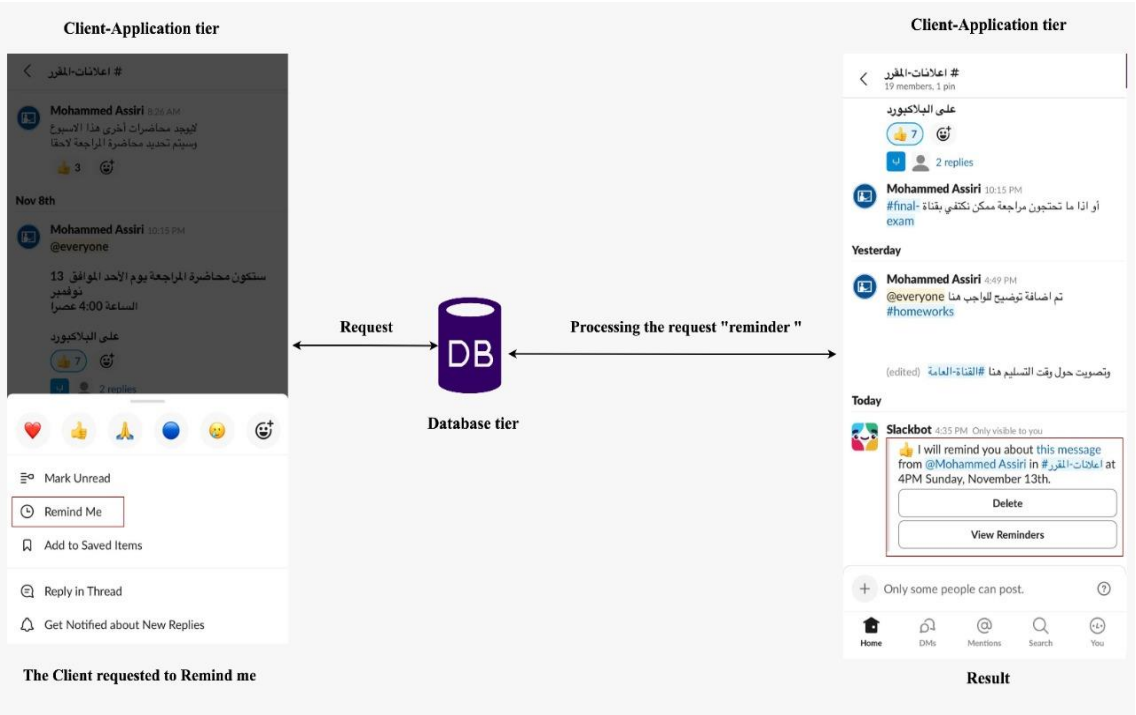
FR 11:" The system shall show the user's status (online or offline)"



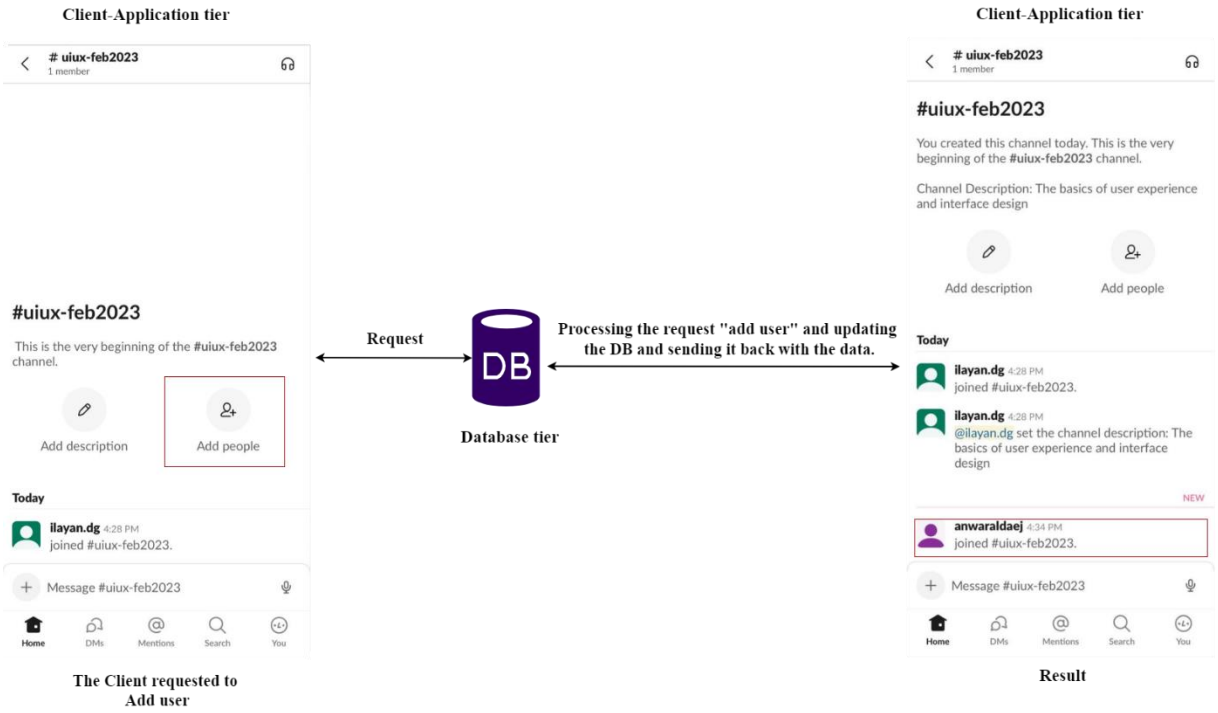
FR 12 :" The system shall provide calls via Wi-Fi"



FR 13 : " The system shall provide the function “remind me" "



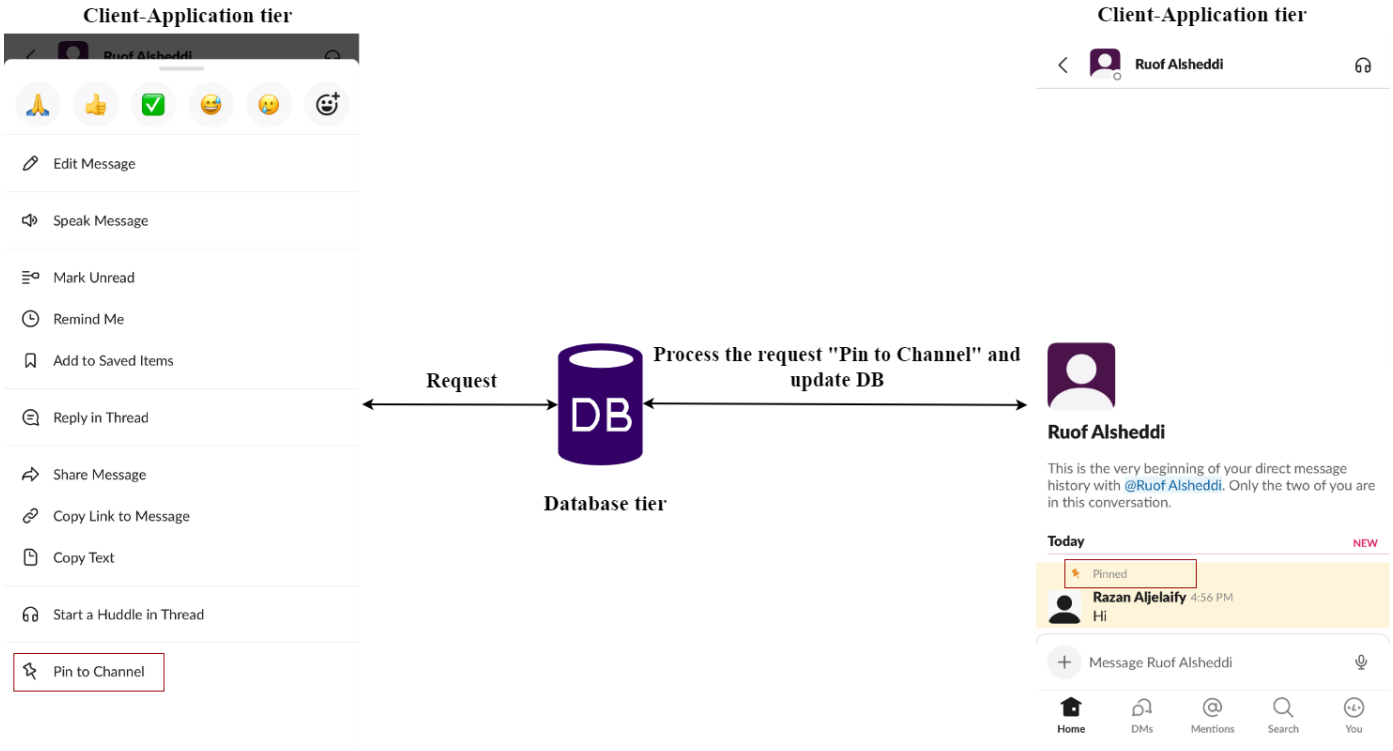
FR 14 : " A user should add other users to the channel."



The Client requested to Add user

Result

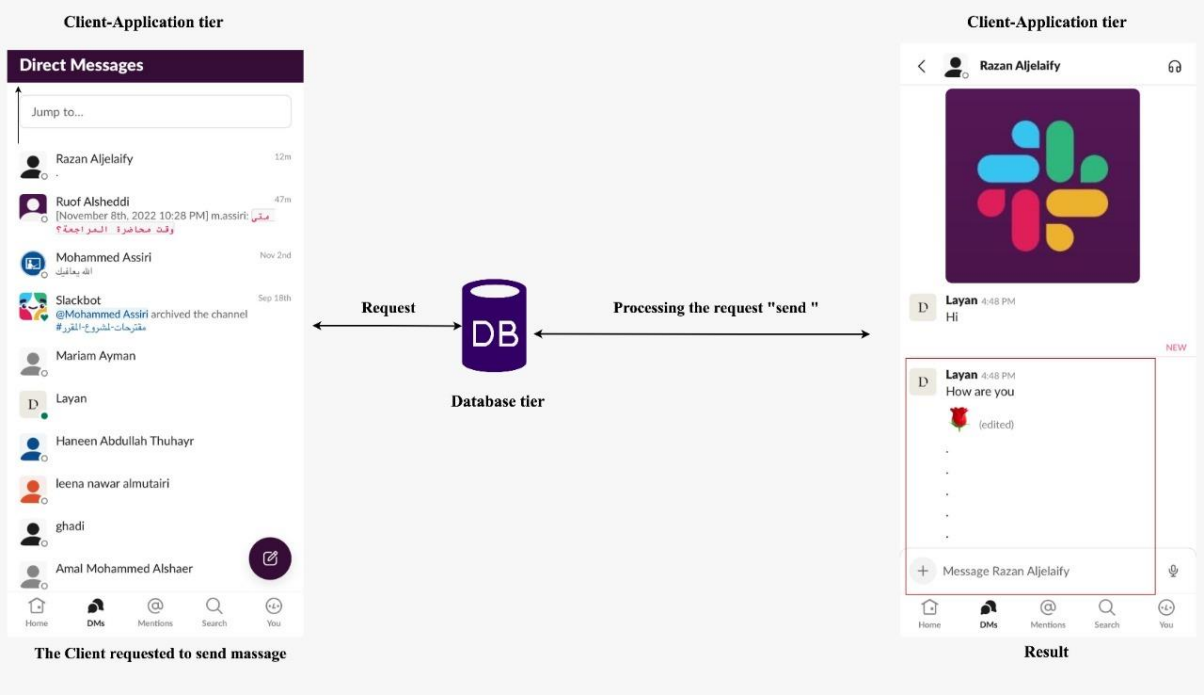
FR 15 :" A user should pin the message to a channel"



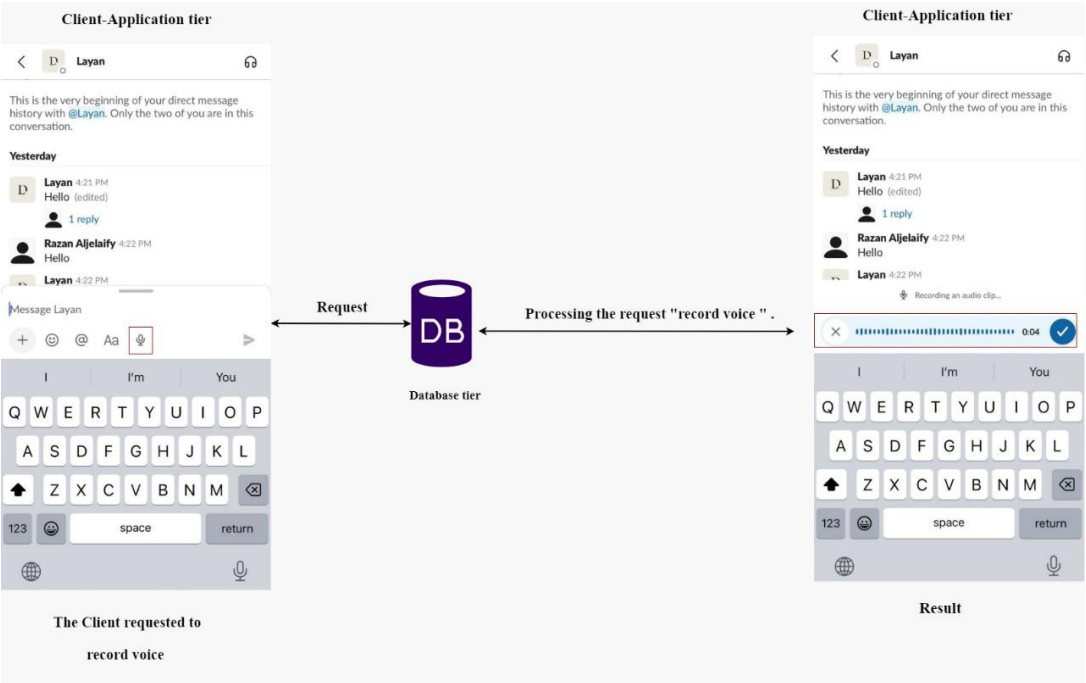
The Client requested to pin the message

Result

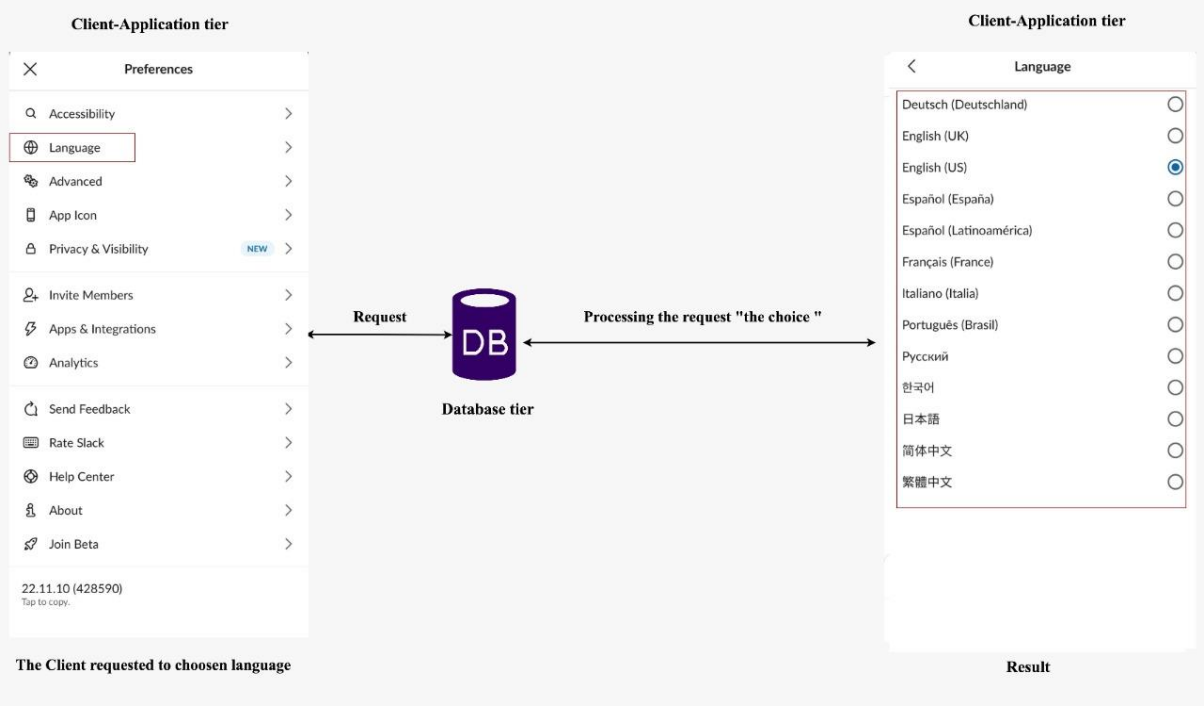
FR 16 : " A user should send a direct message to other users "



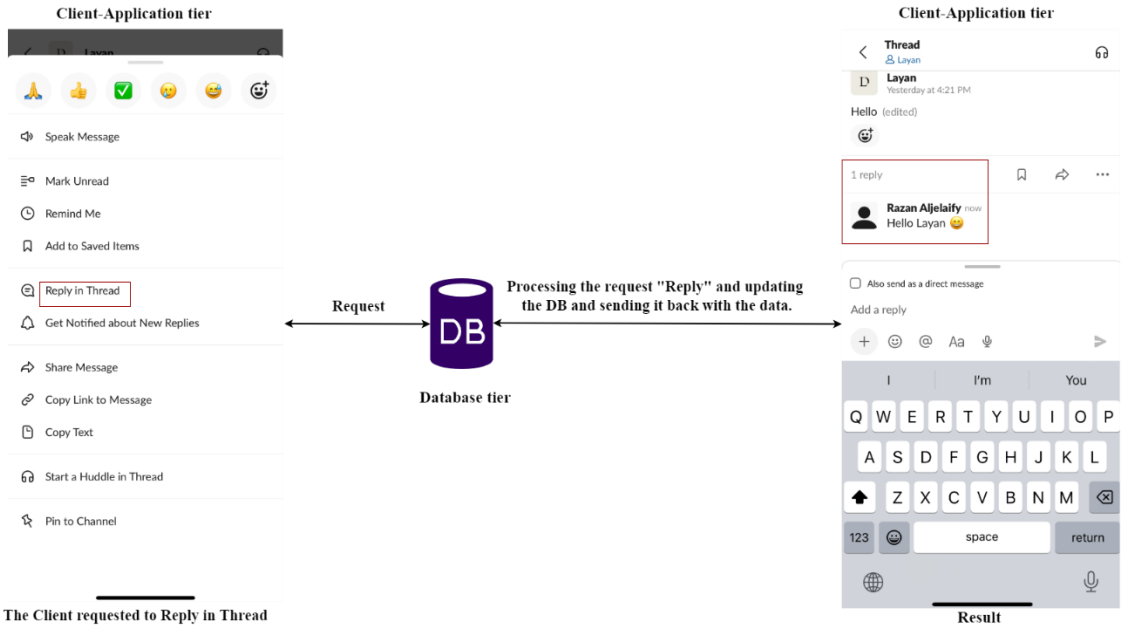
FR 17 : " A user should send a voice note."



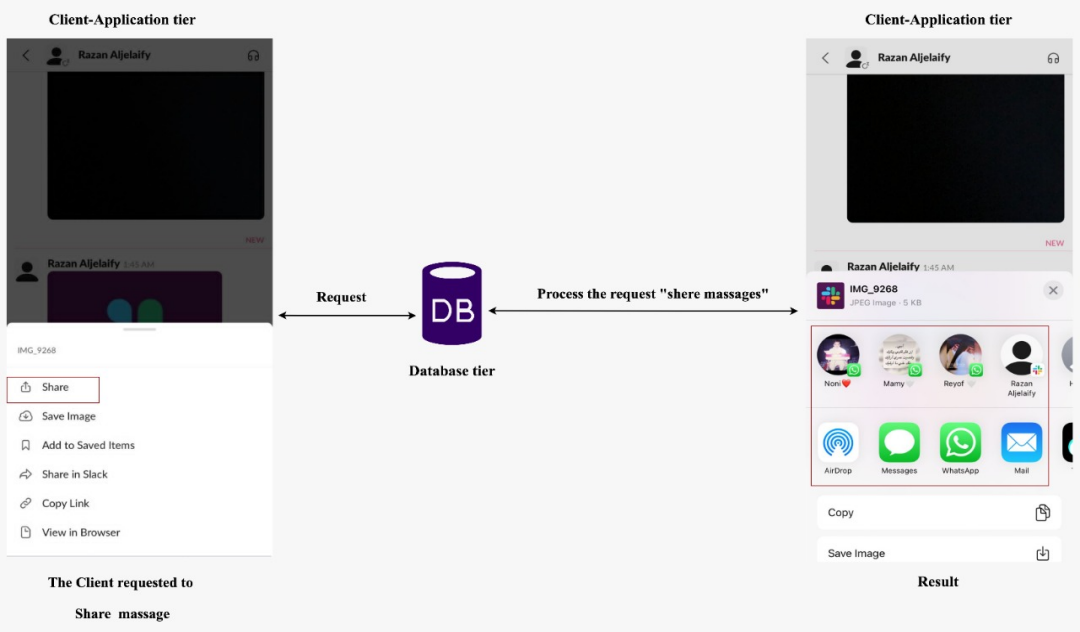
FR 18 :" The system shall provide several languages"



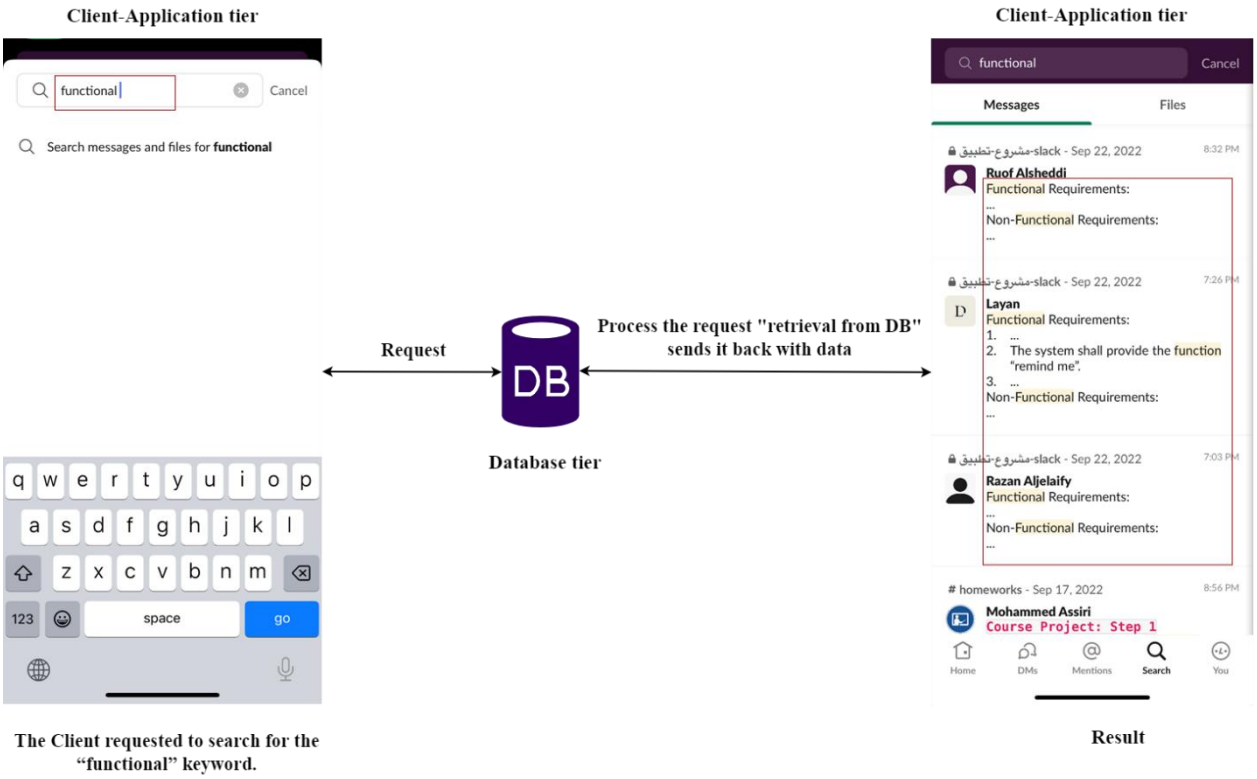
FR 19 : " A user should reply in a thread"



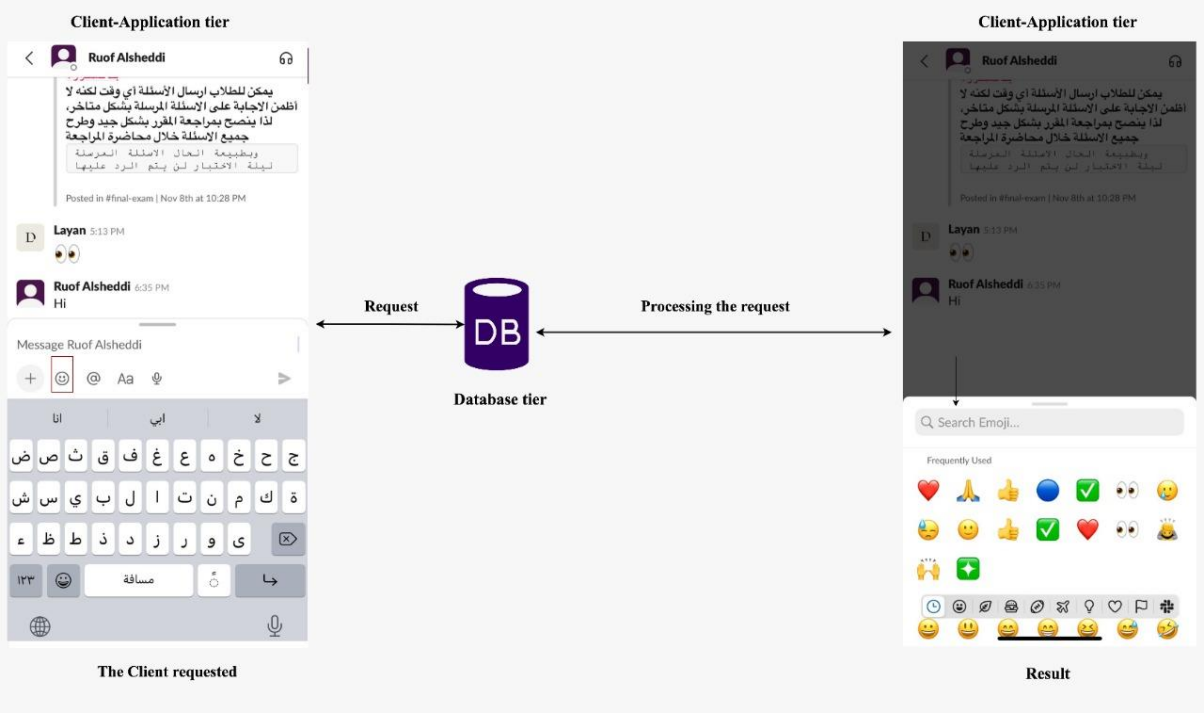
FR 20 : " A user should share messages with other users "



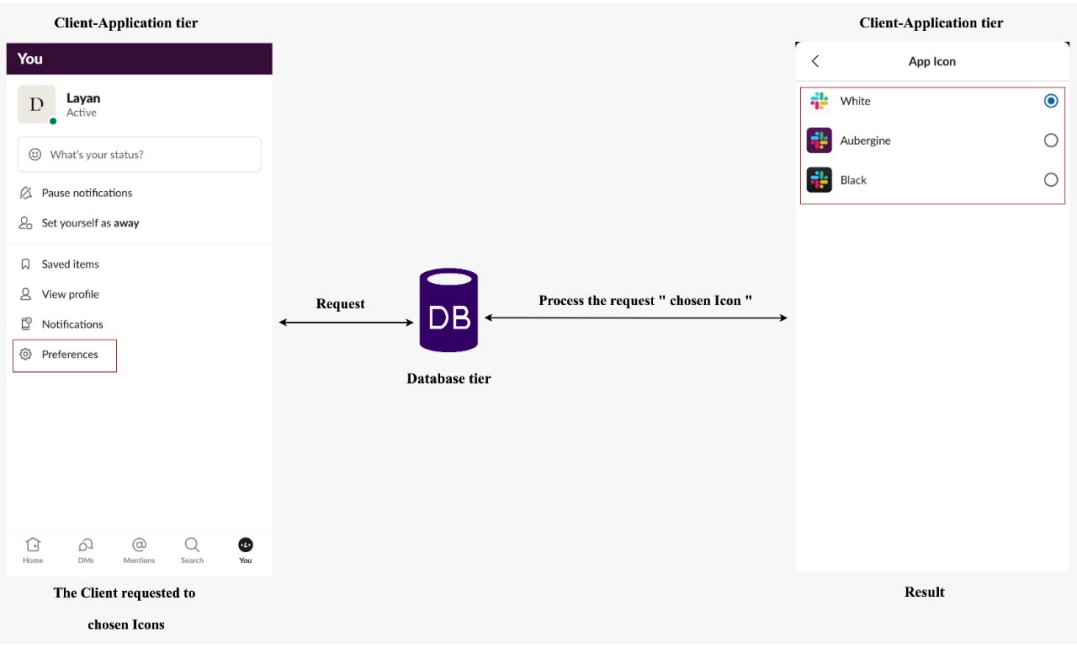
FR 21 : " The system shall provide search fields with keywords."



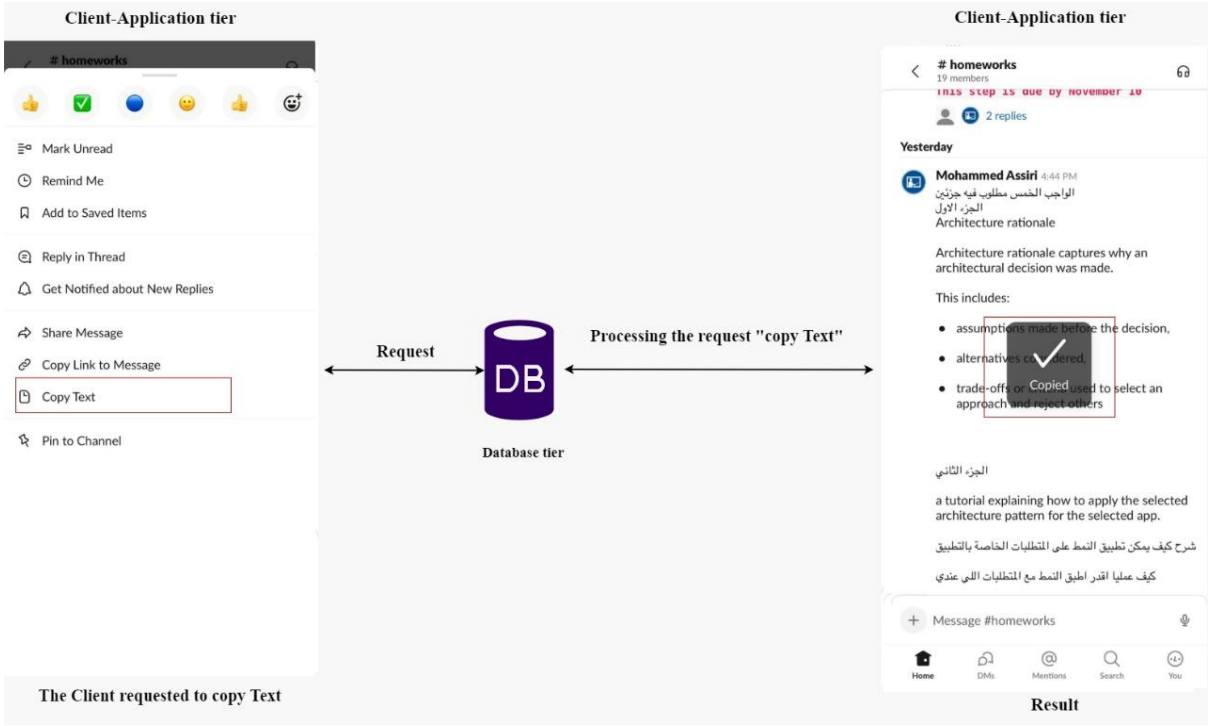
FR 22 :" The system shall provide emojis"



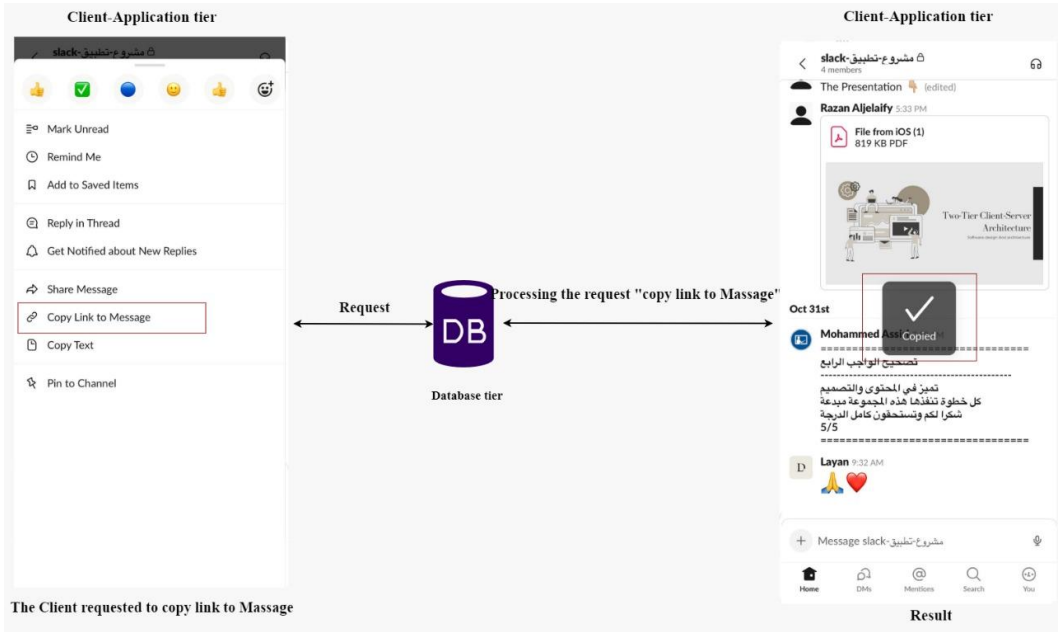
FR 23 :" The system shall provide multiple app icons"



FR 24 :" A user should copy text"

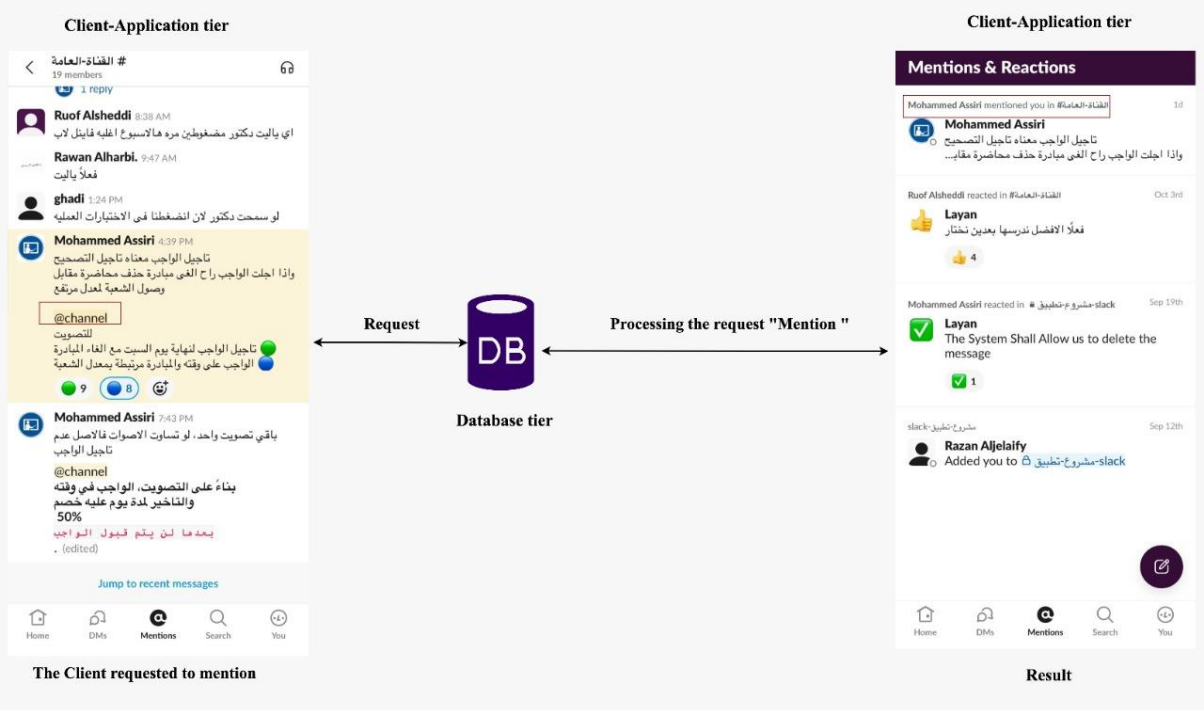


FR 25 :" A user should copy the link to the message"

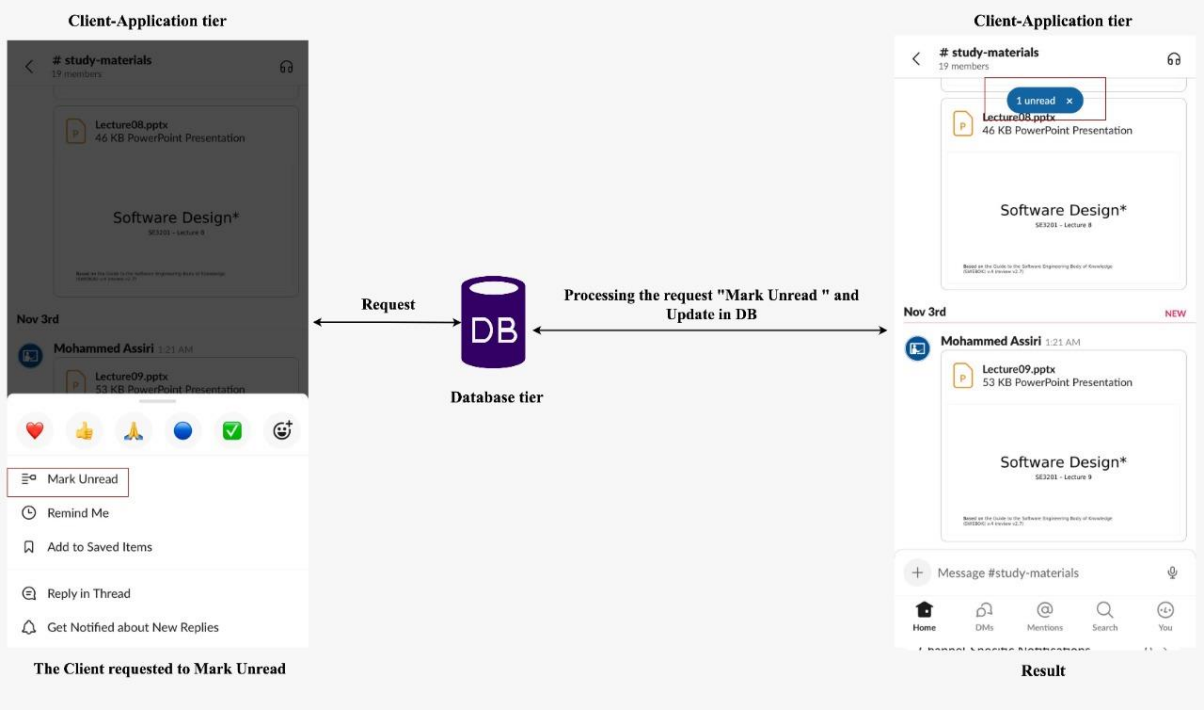


FR 26 :" A user should edit messages after posting."

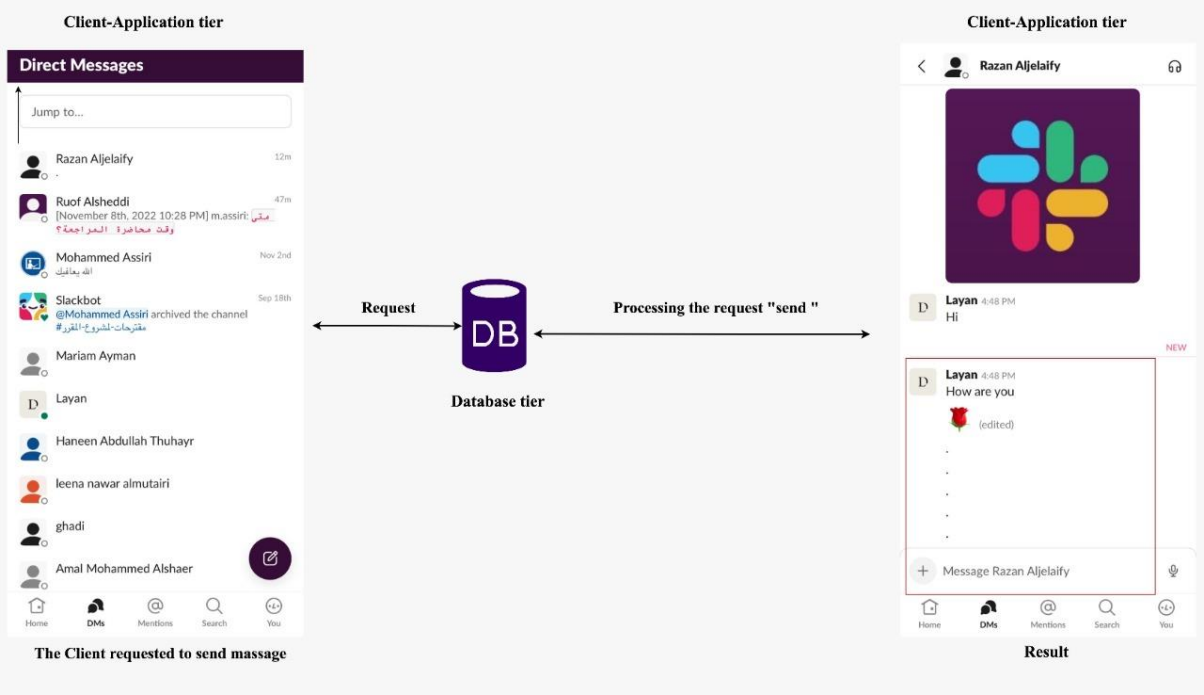
FR 27 :" The system shall support the “mentions” service"



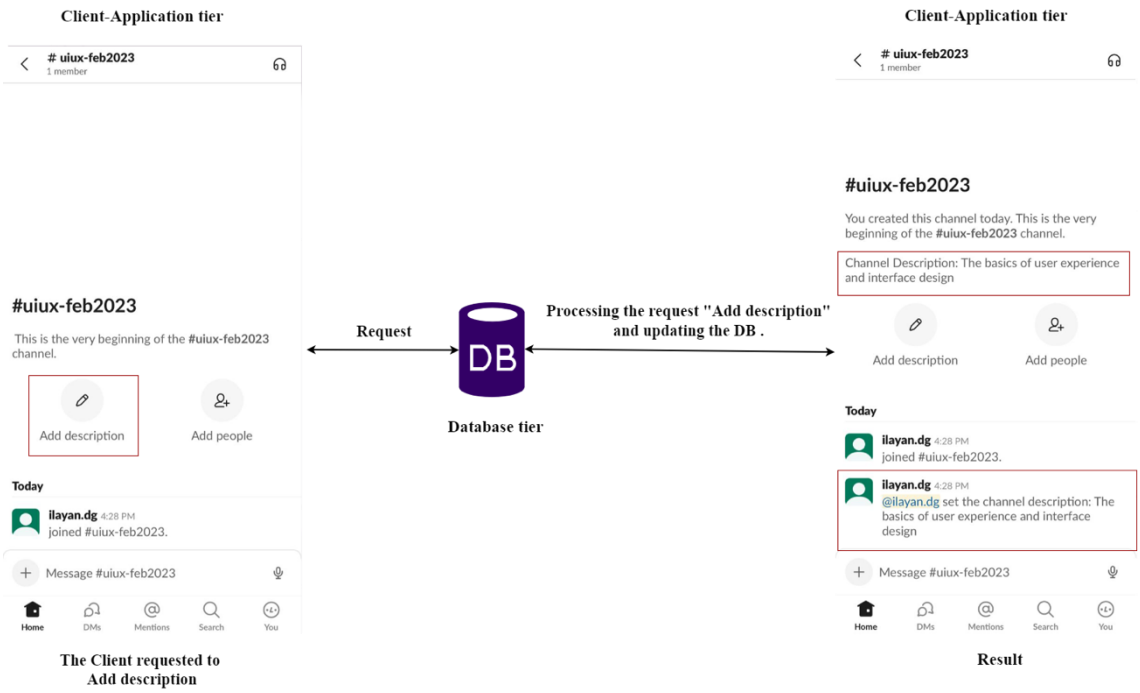
FR 28 :" A user should mark the message Unread."



FR 29 :" A User should send unlimited messages"



FR 30 : " A user should add a description to the channel"



The Client requested to Add description

Result

How is the Non-functional requirement applied in Two-tier client-server architecture?

NFR1:" The user data shall protect using industry-accepted encryption products"

By default, Slack encrypts data at rest and data in transit for all customers. They protect data with tools like Slack Enterprise Key Management (Slack EKM), audit logs, and integrations with top data loss prevention (DLP) providers.

NFR2: "The user shall not be able to use slack services if younger than 16"

This is achieved when the user provides profile information including birth date stored in the database tier and according to that information, the constraints are well set.

NFR3:" All the services shall be available 24/7/365"

We employ two-tier architecture with AWS infrastructure to increase availability.

NFR4: "The system shall use cookies to enable and support security features, and detect malicious activity"

We know that the pattern has a "database tier", which is saved all data that is related to preferences, cookies, and security, and to achieve this NFR there will be a comparison between the saved data, and current data to detect any malicious activity.

NFR5:" The system shall view searching results in 1 second"

All data is saved in the "database tier", and because of the absence of middleware, the result data after searching is transferred fast.

NFR6:" The system shall upload the user's interface within 3 seconds"

The Client-Application processes the request and directly communicates with the database tier to upload the interface fast.

References

- **https://slack.com/help/articles/115004071768-What-is-Slack-**

- **https://medium.com/@fleviankanaiza/two-tier-three-tier-architecture-8b02536d3482**

- **https://www.guru99.com/dbms-architecture.html**

- **Slack engineering website:** https://slack.engineering/how-slack-built-shared-channels/

- **Slack api: https://api.slack.com/rtm**

- **IBM: https://www.ibm.com/cloud/learn/three-tier-architecture**

- **Software architectural patterns in practice: an empirical study:** https://www.researchgate.net/profile/Mohamad-Kassab-2/publication/329605991_Software_architectural_patterns_in_practice_an_empirical_study/links/5d79311992851cacdb32210e/Software-architectural-patterns-in-practice-an-empirical-study.pdf

- **More information about AWS: https://www.eternalsoftsolutions.com/blog/how-to-design-aws-architecture-for-web-application-with-high-availability/**