Log In Join

Back To Module Home

Design Problems

0% completed

Search Module

Mark Module as Completed

High-level Design of WhatsApp

Get introduced to the high-level design of the WhatsApp system.

We'll cover the following

High-level design#

At an abstract level, the high-level design consists of a chat server responsible for communication between the sender and the receiver. When a user wants to send a message to another user, both connect to the chat server. Both users send their messages to the chat server. The chat server then sends the message to the other intended user and also stores the message in the database.

The high-level design of WhatsApp messenger

The following steps describe the communication between both clients:

1. User A and user B create a communication channel with the chat server.
2. User A sends a message to the chat server.
3. Upon receiving the message, the chat server acknowledges back to user A.
4. The chat server sends the message to user B and stores the message in the database if the receiver's status is offline.
5. User B sends an acknowledgment to the chat server.
6. The chat server notifies user A that the message has been successfully delivered.
7. When user B reads the message, the application notifies the chat server.
8. The chat server notifies user A that user B has read the message.

The process is shown in the following illustrations:

Created with Fabric.js 3.6.6

User A and user B create connections with the chat server

1 of 8

API design#

WhatsApp provides a vast amount of features to its users via different APIs. Some features are mentioned below:

- Send message
- Get message or receive message
- Upload a media file or document

Download document or media file

- Send a location
- Send a contact
- Create a status

However, we'll discuss essential APIs related to the first four features.

Send message#

The `sendMessage` API is as follows:

```
sendMessage(sender_ID, reciever_ID, type, text=none, media_object=none,
 document=none)
```

This API is used to send a `text` message from a sender to a receiver by making a POST API call to the `/messages` API endpoint. Generally, the sender's and receiver's IDs are their phone numbers. The parameters used in this API call are described in the following table:

Parameter	Description
`sender_ID`	This is a unique identifier of the user who sends the message.
`reciever_ID`	This is a unique identifier of the user who receives the message.
`type`	The default message `type` is text. This represents whether the sender sends a media file or a document.
`text`	This feild contains the text that has to be sent as a message.
`media_object`	This parameter is defined based on the type parameter. It represents the media file to be sent.
`document`	This represents the document file to be sent.

Get message#

The `getMessage` API is as follows:

```
getMessage(user_Id)
```

Using this API call, users can fetch all unread messages when they come online after being offline for some time.

Parameter	Description
`user_Id`	This is a unique identifier representing the user who has to fetch all unread messages.

Upload media or document file#

The `uploadFile` API is as follows:

```
uploadFile(file_type, file)
```

We can upload media files via the `uploadFile` API by making a POST request to the `/v1/media` API endpoint. A successful response returns an ID that's forwarded to the receiver. The maximum file size for media that can be uploaded is 16 MB, while the limit is 100 MB for a document.

Parameter	Description
`file_type`	This represents the type of file uploaded via the API call.
`file`	This contains the file being uploaded via the API call.

Download a document or media file#

The `downloadFile` API is as follows:

```
downloadFile(user_id, file, file_id)
```

The parameters of this API call are explained in the following table:

Parameter	Description
`user_id`	This is a unique identifier of the user who will download a file.
`file`	This contains the file to be downloaded.
`file_id`	This is a unique identifier of a file. It's generated while uploading a file via `uploadFIle()` API call. The `downloadFile()` API call downloads the media file through this identifier.

In the next lesson, we'll focus on the detailed design of the WhatsApp system.

Requirements of WhatsApp's Design

Next

Detailed Design of WhatsApp

Mark as Completed

Report an Issue