Project Overview

Core Purpose & Scope: The primary goal is to create a messaging application enabling users to register via phone/OTP, manage profiles and contacts, and engage in:

- **One-on-one chats:** Sending text and media.
- **Group chats:** Creation, membership management, admin roles, text and media sharing.
- **Channels:** Creation, subscriber management, broadcasting text and media. Additional in-scope features include media sharing (images, video, files), message status indicators (sent, delivered, read), typing indicators, message history synchronization, basic privacy settings (last seen, profile photo), online status, and basic push notifications.

Initially, features like voice/video calls, dedicated end-to-end encrypted secret chats (though E2EE for standard chats is a consideration), bots, payments, polls, stories, and advanced customization are out of scope.

Product Architecture & User Base: TUASIL will be a standalone, client-server application. The initial focus is on the **backend API** and mobile clients for **iOS and Android**. The target users are general consumers with varying technical proficiency, with intuitive management tools for group/channel administrators.

Key Constraints & Non-Functional Requirements (NFRs): The system must meet stringent requirements for:

- **Performance:** Real-time message delivery (e.g., <500ms), supporting high concurrency (e.g., 100,000 active users/server node).
- **Scalability:** Horizontal scalability for millions to billions of users and massive data volumes (messages, media), necessitating scalable database solutions (e.g., sharding) and media storage.
- **Security:** Robust OTP authentication, encryption of data in transit (TLS 1.2+) and at rest, resilience to common vulnerabilities (OWASP Top 10).
- **Reliability:** High availability (e.g., 99.9% uptime), guaranteed message delivery, and fault tolerance.
- **Usability:** Intuitive UI/UX for typical smartphone users.

Technical Challenges Highlighted: The document, particularly from an expert perspective, identifies significant technical hurdles:

- Implementing efficient **real-time communication** (e.g., WebSockets, MQTT).
- Designing a **scalable backend architecture** (likely microservices).
- **Database scaling** for potentially trillions of messages.
- Complex **state management** for features like online status and read receipts.
- Efficient **media handling**, storage (cloud solutions like S3), and delivery (CDN).
- Implementing robust **security**, with specific mention of the difficulty of correct E2EE implementation.
- Reliable, scalable **push notification** delivery.
- **Offline message synchronization**.

Project Purpose

To be a **real-time messaging platform** that is **designed to emulate the core features of Telegram**. This includes providing functionalities for **secure one-on-one chats, group chats, and channels**, with a primary **emphasis on speed, security, and scalability**.

Stakeholders

1. Users:

- **Description:** This is the primary and most crucial stakeholder group. The SRS defines them as "An individual registered with the system." They are the end-users who will utilize the platform for real-time messaging. The document notes they are "general consumers with varying degrees of technical proficiency." A subset of users also includes "Admins of groups/channels" who have elevated permissions.
- **Involvement/Affected By:** Users are directly affected by all functional and non-functional requirements. They are involved in registration, authentication, contact management, initiating and participating in chats (one-on-one, group, channels), sending and receiving messages (text, media), managing privacy settings, and receiving notifications. Their characteristics and needs drive the usability (NFR-USAB) and performance (NFR-PERF) requirements. Their data security and privacy are paramount (FR-SEC, NFR-SEC).

2. Development Team:

- **Description:** While not explicitly named as a stakeholder group in the "Stakeholders" section (as is sometimes included in SRS), the SRS is intrinsically created for and by the development team. This group includes engineers, designers, and testers responsible for building, implementing, and maintaining the platform. The "Expert Engineer's Perspective" section highlights the technical challenges they will face.
- **Involvement/Affected By:** The development team is involved in translating all functional and non-functional requirements into a working system. They are directly responsible for addressing the technical challenges outlined, including real-time communication, scalable backend/database, media handling, security implementation (TLS/SSL, potentially E2EE), notification integration, and offline synchronization. The maintainability NFRs (NFR-MAINT) directly impact their work.

3. Product Management / Project Management:

- **Description:** This group is responsible for defining the product vision, scope, and roadmap. They likely initiated the creation of this SRS and will oversee the project's progress.
- **Involvement/Affected By:** They are involved in defining the purpose, scope, and overall description of the product (Sections 1 and 2). They use the SRS to ensure the development aligns with the business goals and market needs. Decisions about what is "In Scope" and "Out of Scope" are typically driven by this team.

4. Operations / Infrastructure Team:

- **Description:** This team is responsible for deploying, monitoring, and maintaining the servers, database, and overall infrastructure required to run the messaging platform.
- **Involvement/Affected By:** They are directly impacted by the non-functional requirements related to Performance (NFR-PERF), Scalability (NFR-SCAL), Reliability (NFR-REL), and Security (NFR-SEC), particularly concerning infrastructure security (NFR-SEC-4, NFR-SEC-5). They need to ensure the system can handle high concurrent users, message volume, data storage, and maintain high uptime. The reliance on "Cloud infrastructure" (Section 2.5) is a key consideration for this team.

5. Third-Party Service Providers:

- **Description:** The SRS explicitly mentions dependencies on "Reliable third-party services for SMS delivery (for OTP) and Push Notifications." This group includes companies providing these specific services.
- **Involvement/Affected By:** The project relies on these providers for core functionalities like user authentication via OTP and delivering real-time push notifications (FR-NOTIF). Their reliability and performance directly impact the TUASIL platform's functionality and user experience.

6. Security Experts / Auditors:

- **Description:** This group is concerned with the security posture of the application and its infrastructure. This could be an internal team or external consultants.
- **Involvement/Affected By:** They are directly involved in ensuring that all security requirements (FR-SEC, NFR-SEC) are met. They would review the architecture, implementation, and data handling practices to identify and mitigate vulnerabilities. The mention of resilience to "common web/mobile vulnerabilities (OWASP top 10)" (NFR-SEC-3) is relevant to this group.

7. Legal and Compliance Team (Potentially):

- **Description:** Depending on the target regions and the scale of the platform, a legal and compliance team might become stakeholders, especially concerning data privacy regulations (like GDPR, CCPA, etc.).
- **Involvement/Affected By:** While "Specific regulatory compliance features for all regions" are currently "Out of Scope" (Section 1.2), any future iterations or expansion would involve this team to ensure the platform adheres to relevant laws regarding data collection, storage, and user rights.

Overall Description

1 Product Perspective:

[Your App Name] (TUASIL) is envisioned as a standalone, client-server application. The primary focus for the initial version is on mobile clients for iOS and Android platforms. While a web client and desktop client may be considered in future development, the core requirements and initial implementation prioritize the backend infrastructure and its interaction with mobile devices. The application will integrate with native device capabilities such as contact lists, the camera, and utilize platform-specific push notification services.

2 Product Functions:

The fundamental purpose of TUASIL is to provide users with a robust and secure real-time messaging experience. The system will enable users to perform core functions including:

- Registering and establishing their identity on the platform.
- Managing their personal profile information.
- Finding and organizing their contacts.
- Initiating and participating in various communication formats: one-on-one private chats, group conversations, and broadcast channels.
- Sending and receiving different types of messages, including text and various media formats.
- Configuring and managing their privacy settings to control visibility of their online status, profile information, and who can initiate contact.
- Receiving real-time updates and notifications regarding new messages and other relevant events.

3 User Characteristics:

The intended user base for TUASIL is broad, consisting of general consumers with diverse levels of technical proficiency. The user interface and experience are expected to be intuitive to accommodate this variety. Additionally, the system will support users who take on administrative roles within group chats and channels, requiring specific tools and permissions for managing these communication spaces.

4 Constraints:

The design and implementation of TUASIL are subject to several critical constraints to ensure a high-quality and reliable messaging service:

- **Performance:** Messages must be delivered with minimal delay, aiming for a real-time user experience. The system must be capable of handling a very large volume of concurrent users and the associated message traffic efficiently.
- **Scalability:** The architecture must be inherently scalable, allowing the system to grow horizontally to support a massive increase in the number of users, potentially reaching millions or billions, and the corresponding increase in data volume without significant degradation in performance.
- **Security:** Security is a paramount concern. All data transmitted between clients and servers, as well as data stored at rest, must be secured through encryption and robust authentication mechanisms. Protecting against unauthorized access and data breaches is a fundamental requirement.
- **Reliability:** The platform must maintain high availability and minimize downtime. Message delivery, especially for core messages, must be guaranteed, although eventual consistency may be acceptable for certain features. The system should be resilient to individual component failures.
- **Data Volume:** The nature of a messaging platform means it will generate and handle a massive amount of data, including messages and shared media. [1] Efficient storage, retrieval, and management of this data are critical to the system's performance and scalability.

- **Platform:** The initial development effort is constrained to focusing on the Backend API and native mobile clients for Android and iOS.

25 Assumptions and Dependencies:

The project proceeds based on several key assumptions and dependencies:

- It is assumed that users will have access to a valid phone number for registration and reliable internet connectivity to use the application.
- The project depends on the availability and reliability of third-party services for sending SMS messages (used for OTP during authentication) and for delivering push notifications to mobile devices.
- The project assumes access to a scalable and reliable cloud computing infrastructure, including resources for computation, databases, data storage, and networking.
- The provided database schema is assumed to serve as the foundational data model for the system's backend.

System Requirements

ID	*Category*	*Requirement Description*	*References*
FR-UM-1	User Management	The system SHALL allow a new user to register using a valid phone number.	
FR-UM-2	User Management	The system SHALL send an OTP via SMS to verify the phone number.	
FR-UM-3	User Management	The system SHALL authenticate a user upon successful OTP verification.	
FR-UM-4	User Management	The system SHALL allow a user to set/update their username (must be unique).	users.username
FR-UM-5	User Management	The system SHALL allow a user to set/update their first name, last name, and bio.	users.first_name, last_name, bio
FR-UM-6	User Management	The system SHALL allow a user to upload and set a profile picture.	users.profile_picture_url, potentially media table
FR-UM-7	User Management	The system SHALL display the online status of a user to contacts (based on privacy settings).	users.is_online
FR-UM-8	User Management	The system SHALL update and display the "last seen" timestamp of a user (based on privacy settings).	users.last_seen_at
FR-CM-1	Contact Management	The system SHALL allow a user to add another user as a contact using their phone number.	contacts table
FR-CM-2	Contact Management	The system SHALL allow a user to assign an alias name to a contact.	contacts.alias_name
FR-CM-3	Contact Management	The system SHALL allow a user to view their list of contacts.	
FR-CM-4	Contact Management	The system SHALL allow a user to block another user.	blocked_users table
FR-CM-5	Contact Management	The system SHALL prevent blocked users from initiating contact or seeing online status/profile updates (based on privacy settings).	

FR-CH-1	Chat Management	The system SHALL allow a user to start a private chat with another user.	chats table, chat_participants table
FR-CH-2	Chat Management	The system SHALL allow a user to create a group chat with multiple users.	chats, chat_participants
FR-CH-3	Chat Management	The system SHALL allow a user to manage group chat details (name, picture, description) if they are an admin.	chats.chat_name, chat_picture_url, chat_description, chat_participants.role
FR-CH-4	Chat Management	The system SHALL allow a group admin to add/remove members.	chat_participants
FR-CH-5	Chat Management	The system SHALL allow a group creator/admin to promote/demote other members to admin roles.	chat_participants.role
FR-CH-6	Chat Management	The system SHALL allow a user to leave a group chat.	chat_participants
FR-CH-7	Chat Management	The system SHALL allow a user to create a channel.	chats, chat_participants
FR-CH-8	Chat Management	The system SHALL allow a channel creator/admin to manage channel details (name, picture, description, public link).	chats.chat_name, chat_picture_url, chat_description, public_link, chat_participants.role
FR-CH-9	Chat Management	The system SHALL allow users to subscribe to a channel (if public) or be added (if private).	chat_participants
FR-CH-10	Chat Management	The system SHALL allow a user to delete a chat (private: removes for user, group/channel: removes for user).	chat_participants or soft delete on chats with cascade
FR-MSG-1	Messaging	The system SHALL allow a user to send a text message in a chat.	messages table
FR-MSG-2	Messaging	The system SHALL allow a user to receive text messages in real-time.	
FR-MSG-3	Messaging	The system SHALL allow a user to send media (image, video, file) in a chat, optionally with a caption.	messages, media, messages.content for caption
FR-MSG-4	Messaging	The system SHALL allow a user to receive media messages, with options to view/download.	media
FR-MSG-5	Messaging	The system SHALL indicate the status of a sent message (e.g., single check	

		for sent to server, double check for delivered to recipient's device, colored double check for read in private chats).	
FR-MSG-6	Messaging	The system SHALL display a typing indicator when another user in a private/group chat is typing.	
FR-MSG-7	Messaging	The system SHALL allow a user to reply to a specific message, creating a thread-like visual link.	messages.replied_to_message_id
FR-MSG-8	Messaging	The system SHALL allow a user to forward a message from one chat to another, indicating it was forwarded.	messages.forwarded_from_user_id, forwarded_from_chat_id
FR-MSG-9	Messaging	The system SHALL allow a user to edit their own sent text messages within a defined time window.	messages.edited_at
FR-MSG-10	Messaging	The system SHALL allow a user to delete their own messages (with options to delete for all participants in private/group chats within a time window).	messages.is_deleted for soft delete
FR-MSG-11	Messaging	The system SHALL allow a group/channel admin to delete any message in that chat.	
FR-MSG-12	Messaging	The system SHALL synchronize message history when a user opens a chat or logs in on a new device.	
FR-MSG-13	Messaging	The system SHALL track and display the number of unread messages per chat for each participant.	chat_participants.unread_count, last_read_message_id
FR-MSG-14	Messaging	The system SHALL display a view count for messages in channels.	messages.view_count
FR-MEDIA-1	Media Handling	The system SHALL allow users to upload media files up to a specified size limit.	media
FR-MEDIA-2	Media Handling	The system SHALL store uploaded media securely.	media.file_path_or_url
FR-MEDIA-3	Media Handling	The system SHALL provide URLs or mechanisms for clients to retrieve stored media.	

FR-MEDIA-4	Media Handling	The system SHALL generate thumbnails for images and videos upon upload.	media.thumbnail_url
FR-NOTIF-1	Notifications	The system SHALL send push notifications to a user's active devices for new messages in chats (based on user settings).	user_settings.notifications_private_chats, etc., sessions table
FR-NOTIF-2	Notifications	The system SHALL send push notifications for other events (e.g., new group member, admin change) based on user settings.	
FR-SEC-1	Privacy and Security	The system SHALL authenticate users securely using OTP.	
FR-SEC-2	Privacy and Security	The system SHALL encrypt messages in transit (TLS/SSL).	
FR-SEC-3	Privacy and Security	The system SHALL allow users to configure who can see their phone number.	user_settings.privacy_phone_number
FR-SEC-4	Privacy and Security	The system SHALL allow users to configure who can see their "last seen" status.	user_settings.privacy_last_seen
FR-SEC-5	Privacy and Security	The system SHALL allow users to configure who can see their profile photo.	user_settings.privacy_profile_photo
FR-SEC-6	Privacy and Security	The system SHALL allow users to configure who can add them to groups and channels.	user_settings.privacy_groups_and_channels
FR-SEC-7	Privacy and Security	The system SHALL respect the blocked users list (preventing communication, hiding status based on settings).	

3.2 Non-Functional Requirements (NFRs):

ID	*Category*	*Requirement Description*
NFR-PERF-1	Performance	Messages SHALL be delivered end-to-end within [e.g., 500 ms] under normal network conditions.
NFR-PERF-2	Performance	The system SHALL support [e.g., 100,000] concurrent active users per server node.
NFR-PERF-3	Performance	Retrieving the latest [e.g., 50] messages in a chat SHALL take no longer than [e.g., 200 ms].
NFR-SCAL-1	Scalability	The backend architecture SHALL be horizontally scalable to handle growth in user base and message volume by adding more servers.
NFR-SCAL-2	Scalability	The database SHALL be scalable to handle a growing volume of messages and users, potentially requiring sharding or replication.
NFR-SCAL-3	Scalability	Media storage SHALL be scalable to accommodate terabytes or petabytes of data.
NFR-REL-1	Reliability	The system SHALL have an uptime of at least [e.g., 99.9%].
NFR-REL-2	Reliability	No sent message SHALL be lost after being confirmed as sent by the client.
NFR-REL-3	Reliability	The system SHALL tolerate the failure of a single server node without significant service interruption.
NFR-SEC-1	Security	All data transmitted between client and server SHALL be encrypted using strong protocols (e.g., TLS 1.2+).
NFR-SEC-2	Security	User passwords (if used, though OTP is primary) SHALL be stored using strong hashing algorithms.
NFR-SEC-3	Security	The system SHALL be resilient to common web/mobile vulnerabilities (OWASP top 10).
NFR-SEC-4	Security	Access to sensitive user data in the database SHALL be restricted.
NFR-SEC-5	Security	Data at rest SHALL be encrypted (database, file storage).
NFR-USAB-1	Usability	The user interface SHALL be intuitive and easy to navigate for typical smartphone users.
NFR-USAB-2	Usability	Key actions (sending a message, starting a chat) SHALL be easily discoverable.
NFR-MAINT-1	Maintainability	The codebase SHALL be modular and follow established architectural patterns.
NFR-MAINT-2	Maintainability	Technical documentation for the API and key backend components SHALL be kept up-to-date.
NFR-COMP-1	Compatibility	The Android client SHALL support OS versions [specify range].
NFR-COMP-2	Compatibility	The iOS client SHALL support OS versions [specify range].

Actors

1. **Registered User:** The main type of user who has successfully registered and authenticated with the system.
2. **Group/Channel Admin:** A registered user who has elevated permissions within a specific group chat or channel.
3. **Blocked User:** A registered user who has been blocked by another user.
4. **System (Implicit Actor):** Represents the automated functions of the system, such as sending notifications or OTPs.

1. Registered User:

- **FR-UM-1:** Register using a valid phone number.
- **FR-UM-3:** Authenticate using OTP.
- **FR-UM-4:** Set/update username.
- **FR-UM-5:** Set/update first name, last name, and bio.
- **FR-UM-6:** Upload and set a profile picture.
- **FR-UM-7:** View online status of contacts (based on privacy settings).
- **FR-UM-8:** View "last seen" timestamp of contacts (based on privacy settings).
- **FR-CM-1:** Add another user as a contact.
- **FR-CM-2:** Assign an alias name to a contact.
- **FR-CM-3:** View their list of contacts.
- **FR-CM-4:** Block another user.
- **FR-CH-1:** Start a private chat with another user.
- **FR-CH-2:** Create a group chat.
- **FR-CH-6:** Leave a group chat.
- **FR-CH-7:** Create a channel.
- **FR-CH-9:** Subscribe to a public channel or be added to a private channel.
- **FR-CH-10:** Delete a chat (removes for the user).
- **FR-MSG-1:** Send a text message in a chat.
- **FR-MSG-2:** Receive text messages in real-time.
- **FR-MSG-3:** Send media (image, video, file) in a chat.
- **FR-MSG-4:** Receive media messages (view/download).
- **FR-MSG-7:** Reply to a specific message.
- **FR-MSG-8:** Forward a message to another chat.
- **FR-MSG-9:** Edit their own sent text messages (within a time window).
- **FR-MSG-10:** Delete their own messages (with options to delete for all).
- **FR-MSG-12:** Synchronize message history.
- **FR-MSG-13:** Track unread messages per chat.
- **FR-MEDIA-1:** Upload media files.
- **FR-MEDIA-3:** Retrieve stored media.
- **FR-NOTIF-1:** Receive push notifications for new messages.
- **FR-NOTIF-2:** Receive push notifications for other events (based on settings).
- **FR-SEC-3:** Configure who can see their phone number.
- **FR-SEC-4:** Configure who can see their "last seen" status.
- **FR-SEC-5:** Configure who can see their profile photo.
- **FR-SEC-6:** Configure who can add them to groups and channels.

2. Group/Channel Admin:

- **FR-CH-3:** Manage group chat details (name, picture, description).
- **FR-CH-4:** Add/remove members from a group chat.
- **FR-CH-5:** Promote/demote other members to admin roles in a group chat.
- **FR-CH-8:** Manage channel details (name, picture, description, public link).
- **FR-MSG-11:** Delete any message in a group chat or channel they administer.

3. Blocked User:

- **FR-CM-5:** Be prevented from initiating contact with the user who blocked them.
- **FR-CM-5:** Have their online status/profile updates hidden from the user who blocked them (based on privacy settings).

4. System (Implicit Actor):

- **FR-UM-2:** Send an OTP via SMS.
- **FR-MSG-5:** Indicate the status of a sent message.
- **FR-MSG-6:** Display a typing indicator.
- **FR-MSG-14:** Display a view count for messages in channels.
- **FR-MEDIA-2:** Store uploaded media securely.
- **FR-MEDIA-4:** Generate thumbnails for media.
- **FR-NOTIF-1:** Send push notifications for new messages.
- **FR-NOTIF-2:** Send push notifications for other events.
- **FR-SEC-1:** Authenticate users securely.
- **FR-SEC-2:** Encrypt messages in transit.
- **FR-SEC-7:** Respect blocked users list (enforce restrictions).

Domain Requirements

Domain Requirements in an SRS describe the characteristics, constraints, and rules that are inherent to the specific business or problem domain the software is addressing. They reflect the vocabulary, concepts, and fundamental truths of that domain.

While the provided SRS for "TUASIL - A Secure Messaging Platform" doesn't have a separate section explicitly titled "Domain Requirements," these requirements are woven throughout the document, particularly in the definitions, the overall description, and the functional requirements. They define the core concepts and rules of a secure messaging domain.

Here are the key aspects of the Domain Requirements as extracted from the SRS:

1. Core Domain Concepts:

- **User:** A fundamental entity, defined as "An individual registered with the system." Users have associated attributes like phone number, username, first name, last name, bio, profile picture, online status, and last seen timestamp. (Ref: FR-UM-x, Section 1.3)
- **Chat:** A central concept representing a conversation thread. This domain concept is specialized into different types: Private Chat (one-on-one), Group Chat (multiple users), and Channel (broadcasting

from admins to subscribers). Chats have attributes like name, picture, and description. (Ref: FR-CH-x, Section 1.3)
- **Message:** The basic unit of communication within a chat. Messages have content (text, media), status (sent, delivered, read), and can be replied to or forwarded. They are associated with a sender and a chat. (Ref: FR-MSG-x, Section 1.3)
- **Contact:** Represents a relationship between users, allowing one user to add another. Aliases can be assigned to contacts. (Ref: FR-CM-x)
- **Channel:** A specific type of chat where communication is primarily one-way from administrators to subscribers. Channels can be public or private. (Ref: FR-CH-7, FR-CH-8, FR-CH-9, FR-MSG-14)
- **Group Chat:** A type of chat allowing multi-way communication among invited members. Groups have administrators with specific management privileges. (Ref: FR-CH-2, FR-CH-3, FR-CH-4, FR-CH-5, FR-CH-6, FR-MSG-11)
- **Media:** Represents non-text content shared within messages, such as images, videos, and files. Media has attributes like file path/URL and thumbnails. (Ref: FR-MEDIA-x, Section 1.3)

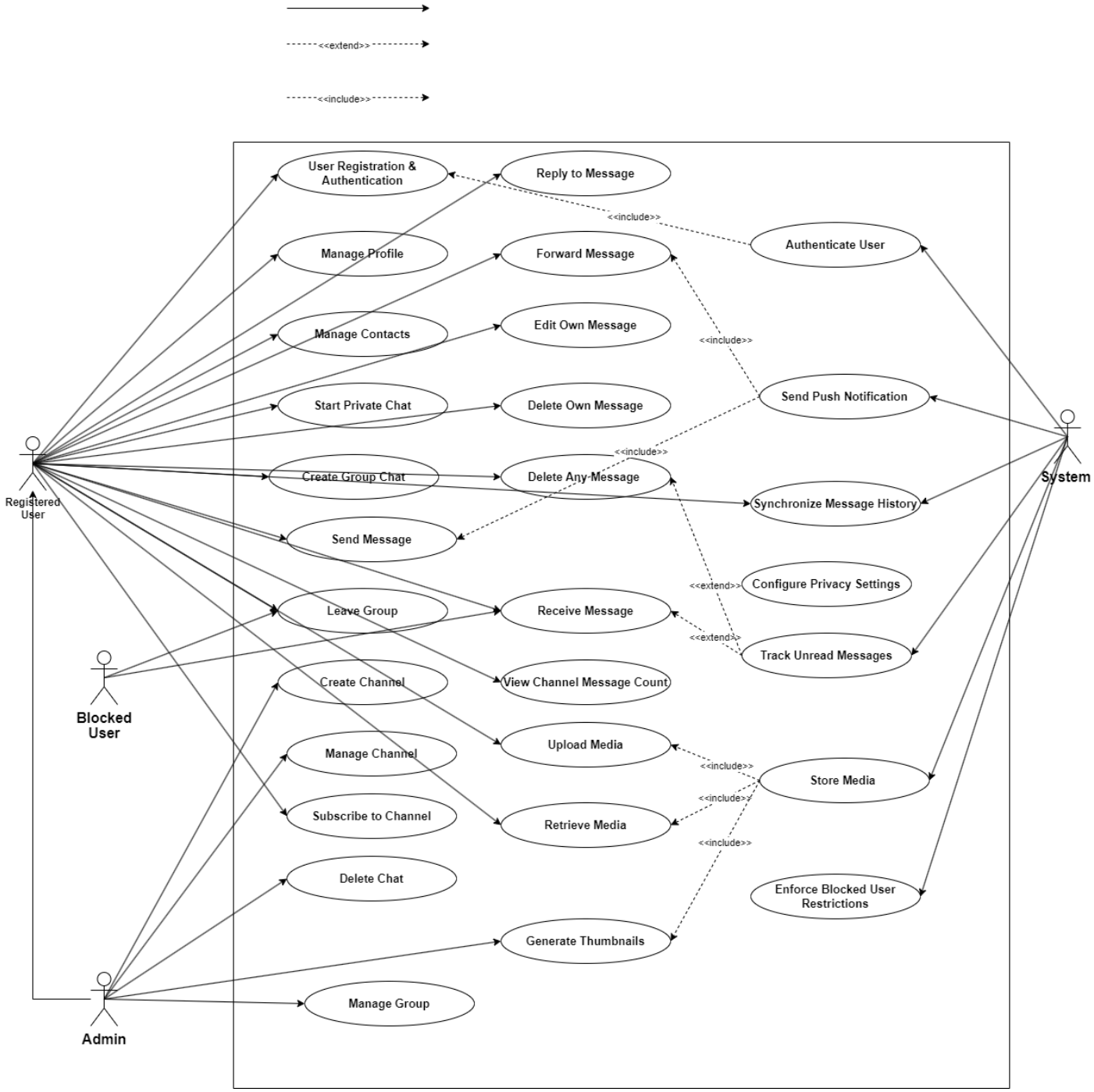
2. Domain Rules and Policies:

- **User Identification:** Users are primarily identified and authenticated via their phone number using an OTP mechanism. (Ref: FR-UM-1, FR-UM-2, FR-UM-3, FR-SEC-1)
- **Contacting Users:** Users can add contacts using phone numbers. Blocking a user prevents communication and hides certain status information based on privacy settings. (Ref: FR-CM-1, FR-CM-4, FR-CM-5, FR-SEC-7)
- **Chat Membership and Roles:** Private chats are between two users. Group chats and channels have explicit membership management, including the concept of administrator roles with specific permissions (adding/removing members, managing chat details, deleting messages). (Ref: FR-CH-x)
- **Message Delivery and Status:** Messages are intended for real-time delivery. The system tracks and indicates the status of messages (sent, delivered, read) and provides features like typing indicators and view counts for channels. (Ref: FR-MSG-2, FR-MSG-5, FR-MSG-6, FR-MSG-14)
- **Message Immutability and Mutability:** While core message history is synchronized, users have limited ability to edit or delete their own messages within a certain time window. Admins have broader message deletion privileges in groups/channels. (Ref: FR-MSG-9, FR-MSG-10, FR-MSG-11, FR-MSG-12)
- **Privacy Settings:** The domain includes concepts of user privacy regarding phone number visibility, last seen status, profile photo, and the ability to be added to groups/channels. These settings define how user information is shared within the network. (Ref: FR-SEC-3, FR-SEC-4, FR-SEC-5, FR-SEC-6)

3. Domain Constraints:

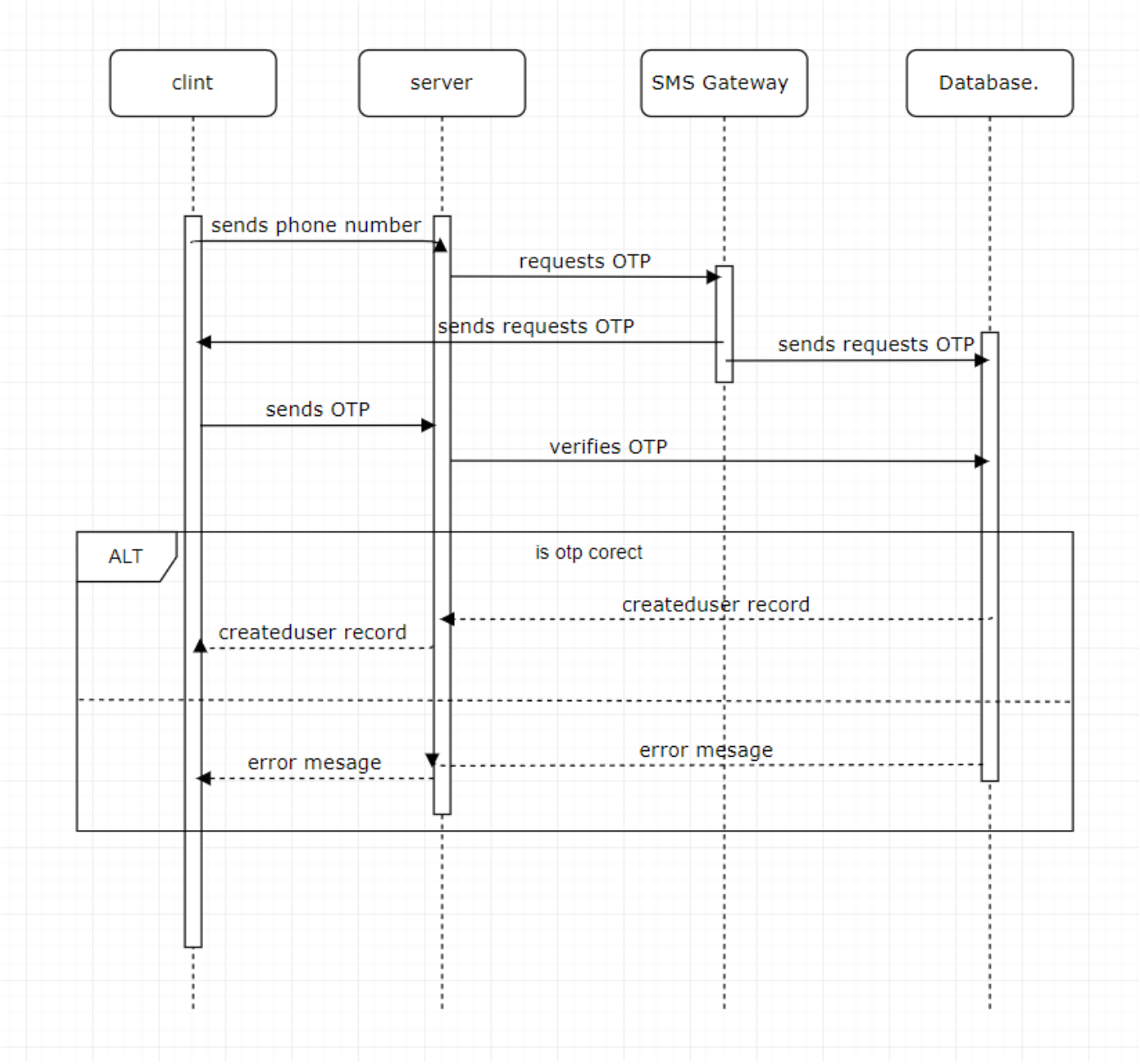
- **Real-time Communication:** A core constraint of the messaging domain is the expectation of real-time or near real-time message delivery. (Ref: NFR-PERF-1)
- **Data Volume:** Messaging platforms inherently deal with a massive and ever-growing volume of data (messages, media), which is a significant domain constraint impacting storage and retrieval design. (Ref: Section 2.4, NFR-SCAL-2, NFR-SCAL-3)
- **Security and Privacy:** Due to the sensitive nature of personal communication, robust security (encryption, authentication, data protection) and user privacy controls are fundamental constraints of this domain. (Ref: FR-SEC-x, NFR-SEC-x, Section 2.4)

Use cas


<<extend>>
<<include>>
User Registration & Authentication
Reply to Message
<<include>>
Authenticate User
Manage Profile
Forward Message
Manage Contacts
Edit Own Message
<<include>>
Start Private Chat
Delete Own Message
Send Push Notification
<<include>>
Create Group Chat
Delete Any Message
Synchronize Message History
Send Message
<<extend>>
Configure Privacy Settings
Leave Group
Receive Message
<<extend>>
Track Unread Messages
Create Channel
View Channel Message Count
Upload Media
<<include>>
Store Media
Manage Channel
<<include>>
Subscribe to Channel
Retrieve Media
<<include>>
Delete Chat
Enforce Blocked User Restrictions
Generate Thumbnails
Manage Group
Registered User
Blocked User
Admin
System

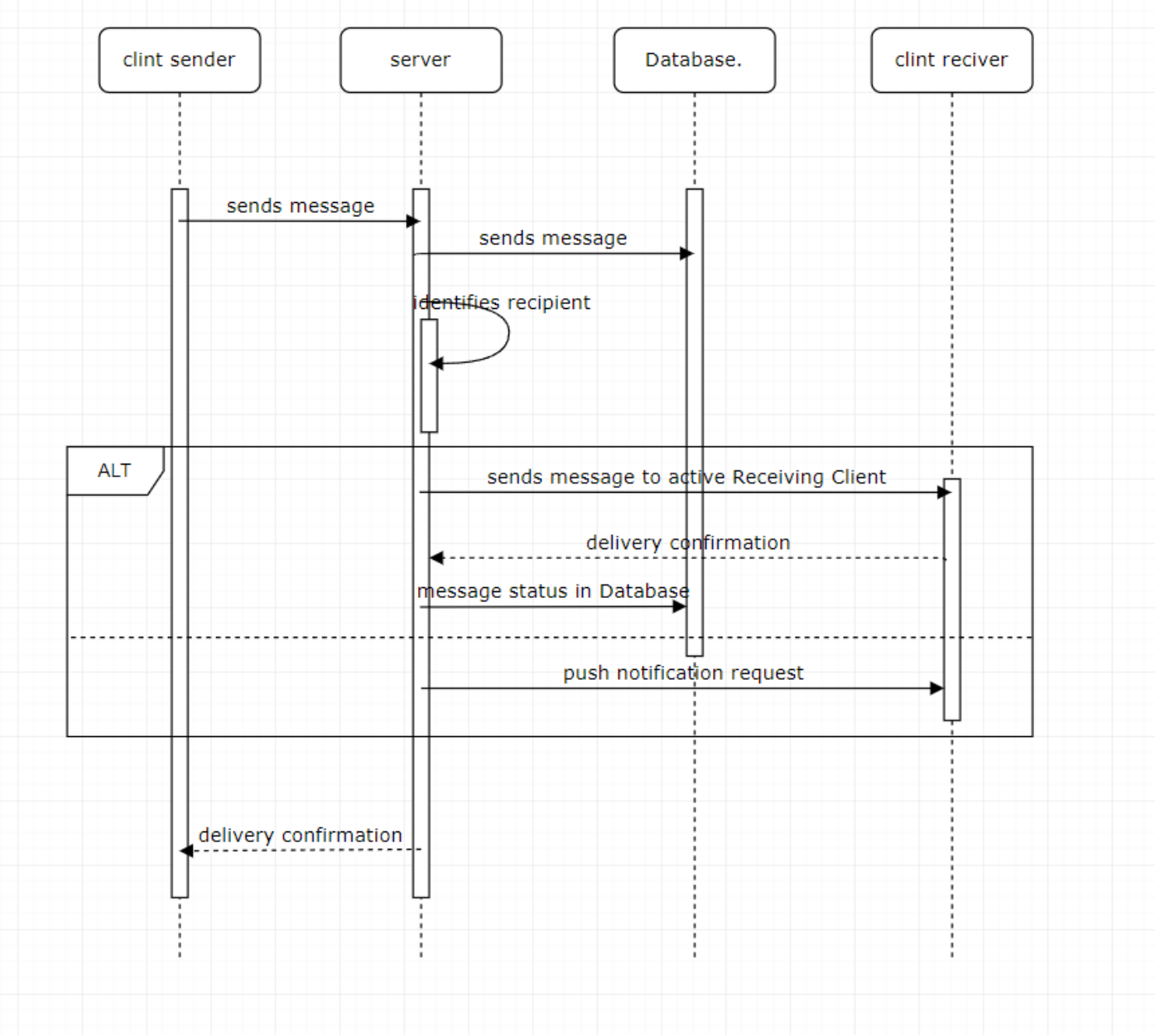
sequence diagrams

- **User Registration and Authentication:**
    - **Purpose:** To show the step-by-step process of a new user signing up and verifying their identity.
    - **Participants:** Mobile Client, Backend (Authentication Service), SMS Gateway (Third-Party), Database.
    - **Flow:** User enters phone number -> Client sends number to Backend -> Backend requests OTP from SMS Gateway -> SMS Gateway sends OTP to phone -> User enters OTP in Client -> Client sends OTP to Backend -> Backend verifies OTP with Database -> Backend creates user record and sends success/auth token to Client.



- **Sending a Text Message (Private Chat):**

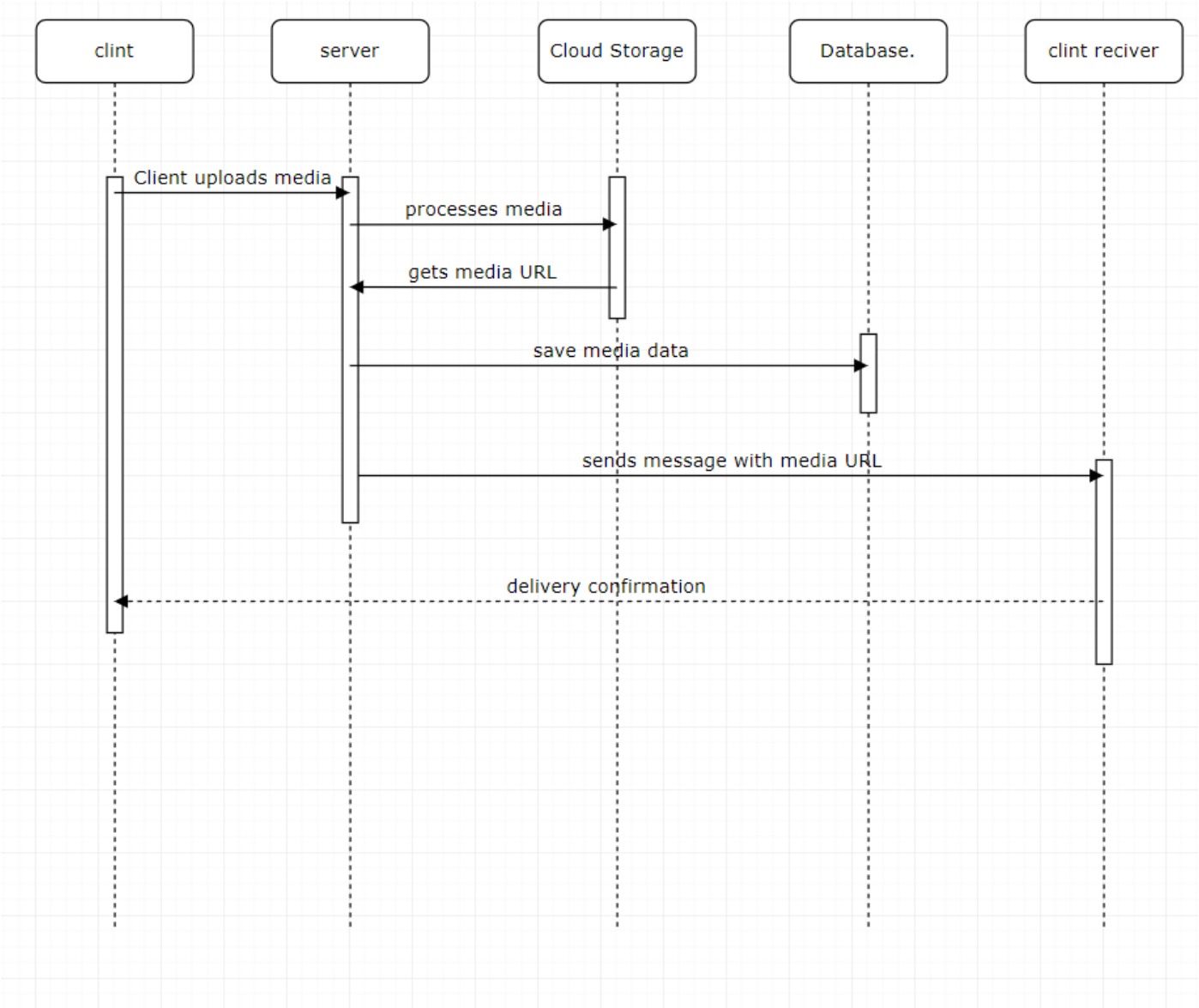
- **Purpose:** To detail the flow of sending a simple text message from one user to another in real-time.
- **Participants:** Sending Mobile Client, Backend (Messaging Service, Notification Service), Database, Receiving Mobile Client.
- **Flow:** Sender types and sends message -> Sending Client sends message to Backend -> Backend stores message in Database -> Backend identifies recipient(s) -> Backend sends message to active Receiving Client(s) -> Backend sends push notification request to Notification Service -> Receiving Client receives message -> Receiving Client sends delivery confirmation to Backend -> Backend updates message status in Database.



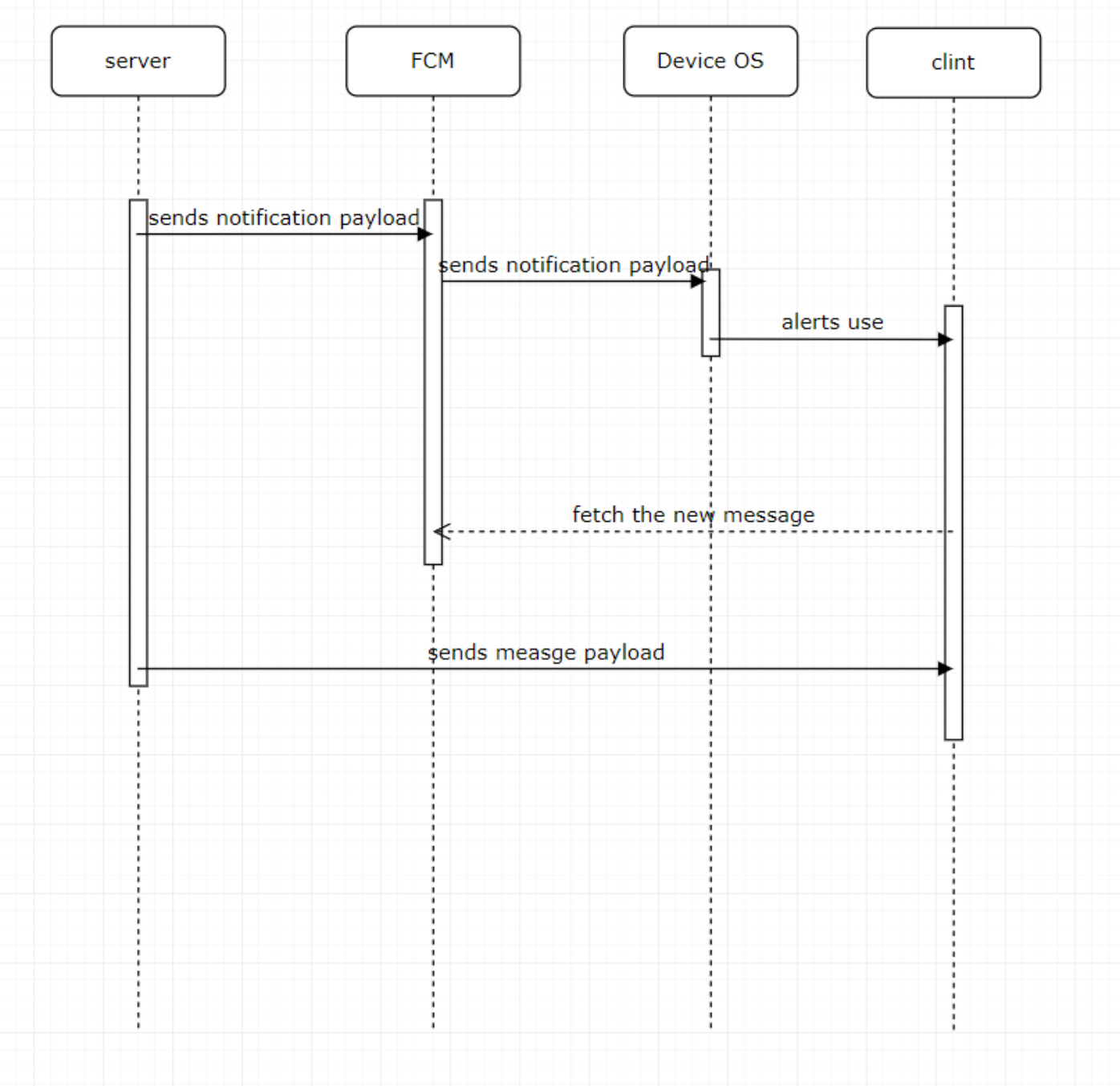
- **Sending a Media Message:**

  - **Purpose:** To show the process of uploading and sending a file (image, video, etc.) with a message.
  - **Participants:** Sending Mobile Client, Backend (Media Service, Messaging Service, Notification Service), Cloud Storage (Third-Party), Database, Receiving Mobile Client.
  - **Flow:** Sender selects media -> Sending Client uploads media to Backend/Cloud Storage -> Backend Media Service processes media (e.g., generates thumbnail) -> Backend gets media URL -> Sending

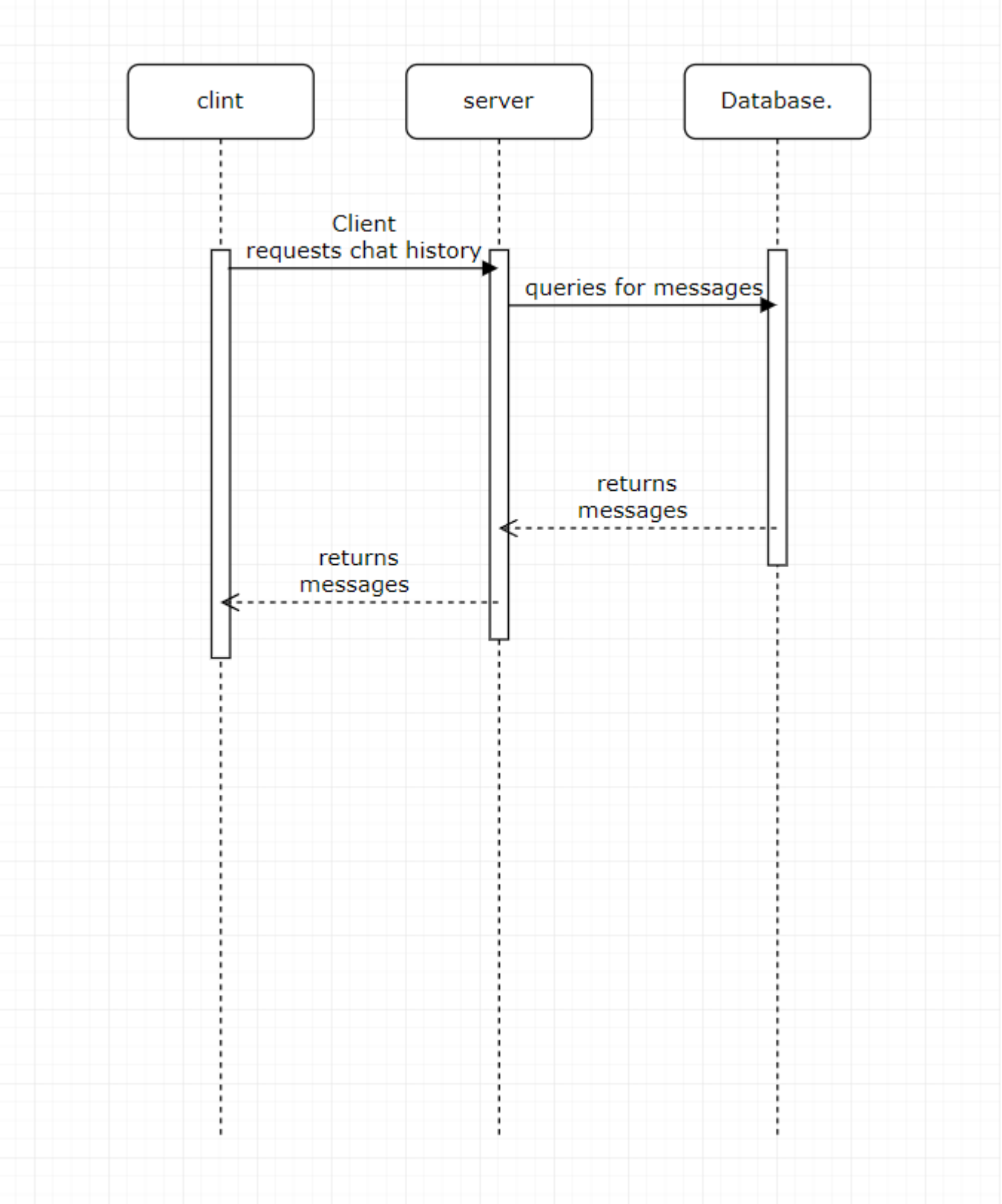
Client sends message with media URL to Backend Messaging Service -> (Follows steps similar to Sending Text Message, but message content includes media reference).



- **Receiving a Message (Push Notification Flow):**
    - **Purpose:** To specifically illustrate how the system uses push notification services to alert offline or backgrounded users.
    - **Participants:** Backend (Notification Service), APNS (Apple Push Notification Service) / FCM (Firebase Cloud Messaging), Receiving Mobile Device OS, Receiving Mobile Client.
    - **Flow:** Backend Notification Service sends notification payload to APNS/FCM -> APNS/FCM delivers notification to device OS -> Device OS alerts user and/or wakes up Receiving Mobile Client -> Receiving Client connects to Backend to fetch the new message content.

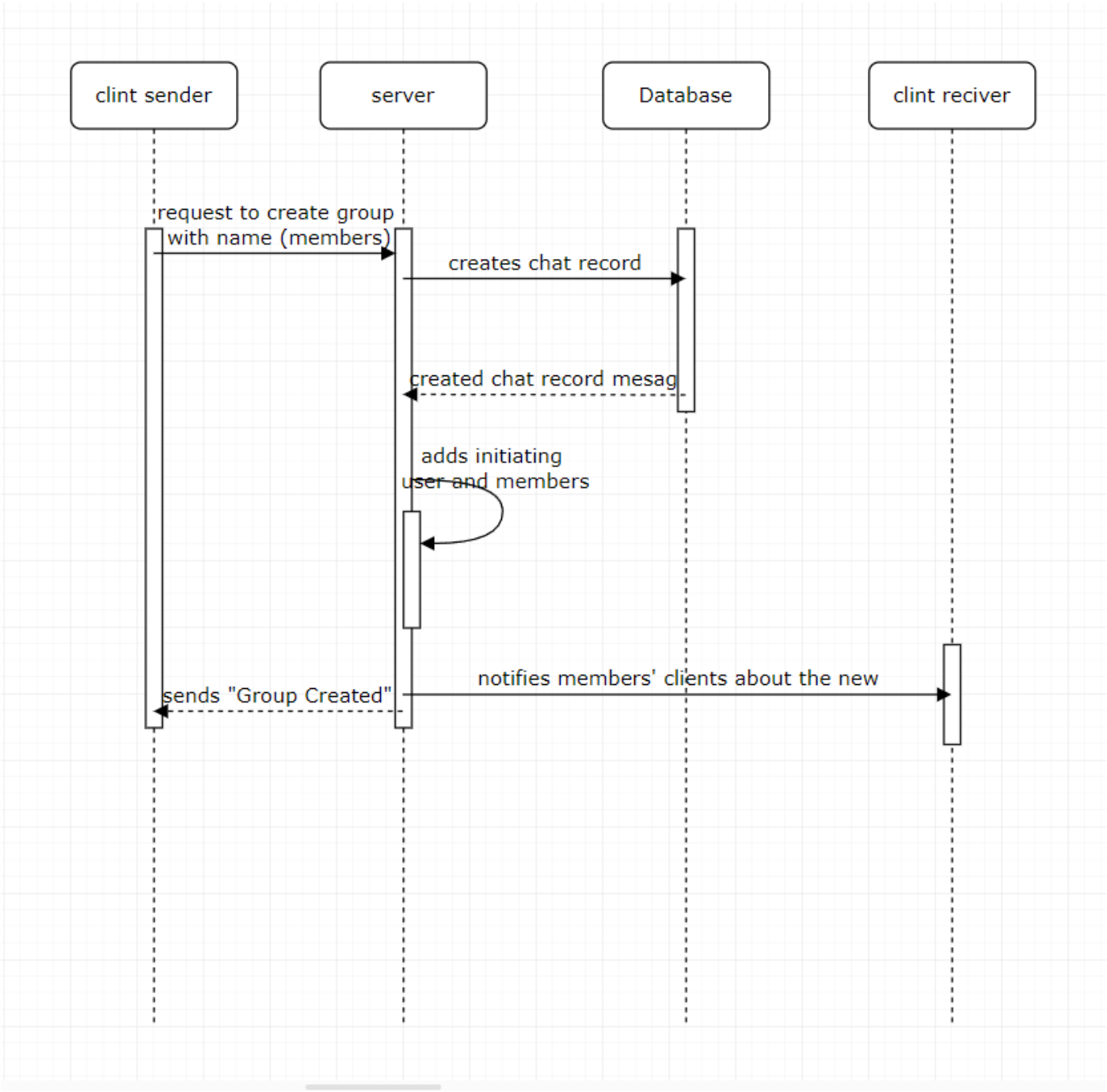


- **Loading Chat History5:**
    - **Purpose:** To show how a client retrieves past messages when opening a chat or synchronizing history.
    - **Participants:** Mobile Client, Backend (Messaging Service), Database.
    - **Flow:** Client requests chat history (e.g., latest 50 messages from a certain point) for a specific chat ID from Backend -> Backend queries Database for messages -> Database returns messages to Backend -> Backend sends messages to Client -> Client displays messages.

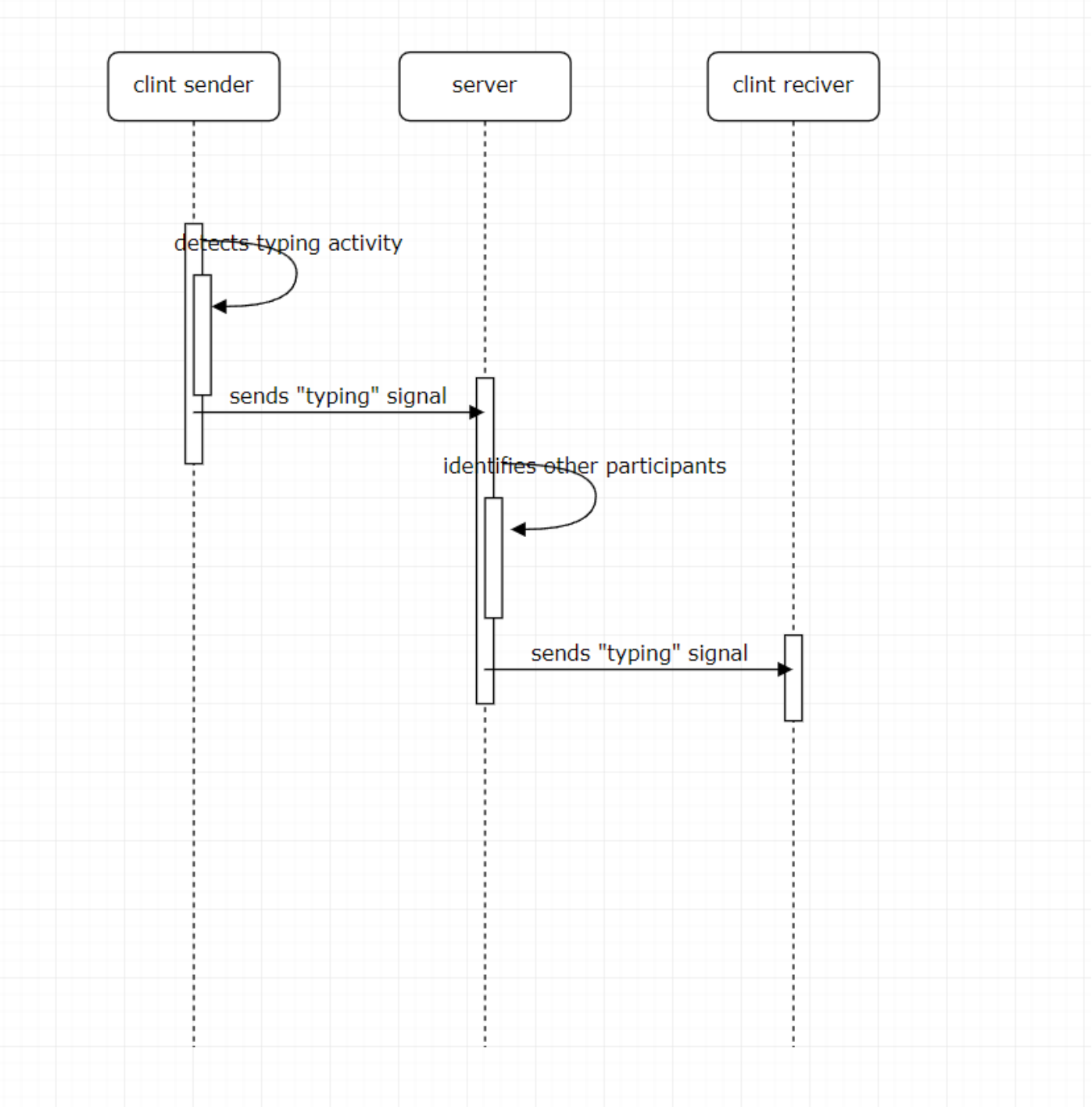


- **Creating a Group Chat:**
    - **Purpose:** To illustrate the steps involved in a user creating a new group and adding initial members.
    - **Participants:** Initiating Mobile Client, Backend (Chat Service, Messaging Service), Database, Members' Mobile Clients (optional, for immediate notification).
    - **Flow:** Initiating Client sends request to create group with name, members list to Backend -> Backend Chat Service creates chat record in Database -> Backend adds initiating user and members

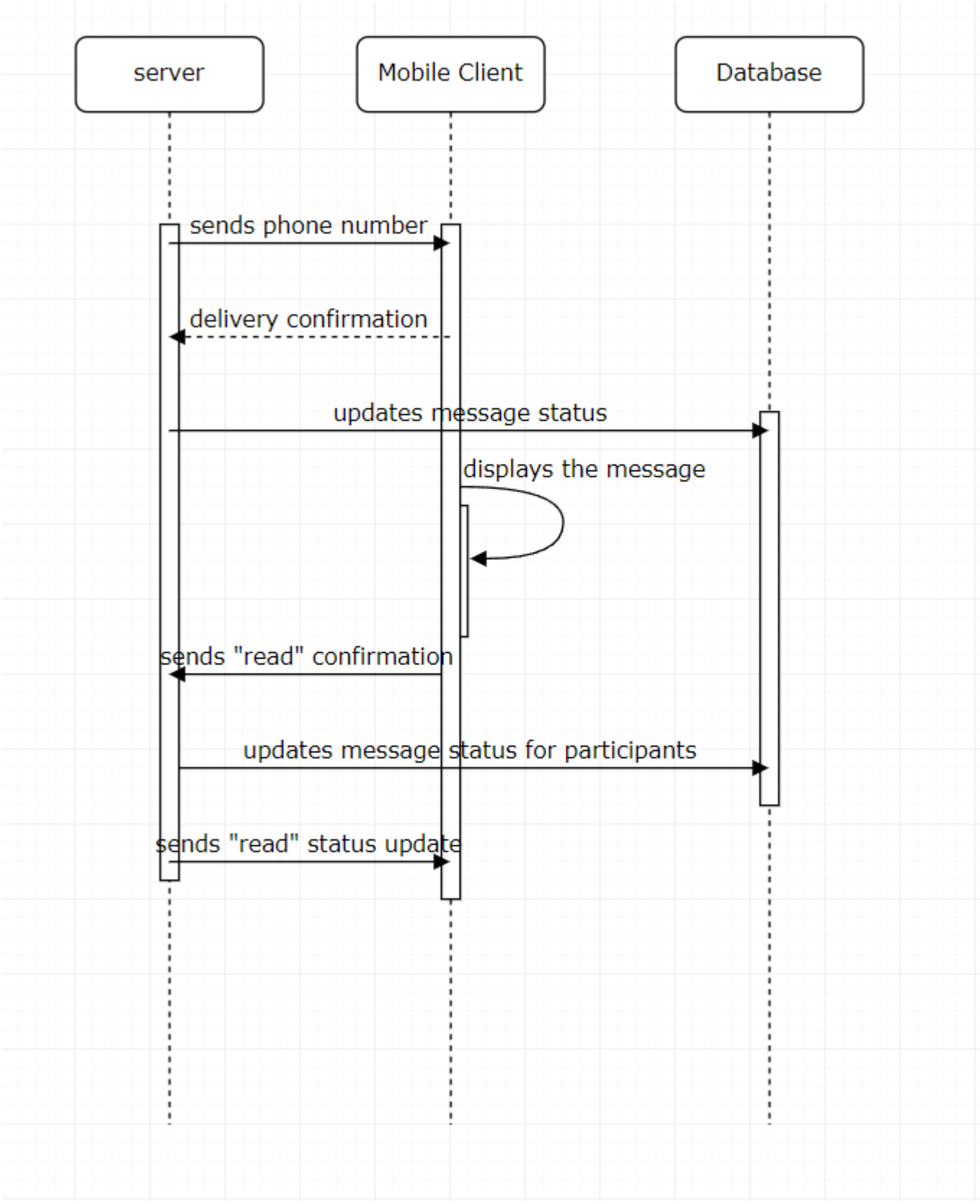
to chat_participants in Database -> Backend sends "Group Created" system message to the new chat -> (Optionally) Backend notifies members' clients about the new group.



- **Sending Typing Indicator:**
    - **Purpose:** To show the lightweight, real-time signaling of a user typing.
    - **Participants:** Typing Mobile Client, Backend (Presence/Messaging Service), Other Participants' Mobile Clients.
    - **Flow:** Typing Client detects typing activity -> Typing Client sends "typing" signal to Backend -> Backend identifies other participants in the chat -> Backend sends "typing" signal to other participants' active Clients -> Other Clients display typing indicator.



- **Updating Message Status (Delivered/Read):**
    - **Purpose:** To show how read receipts and delivery confirmations are handled.
    - **Participants:** Receiving Mobile Client, Backend (Messaging Service), Database, Sending Mobile Client.
    - **Flow:** Receiving Client receives a message -> Receiving Client sends "delivered" confirmation to Backend -> Backend updates message status in Database -> (Later) Receiving Client displays the message -> Receiving Client sends "read" confirmation to Backend -> Backend updates message status (e.g., `last_read_message_id` for the user in `chat_participants`) and potentially the message status in the `messages` table -> Backend sends "read" status update to Sending Mobile Client.


server
Mobile Client
Database
sends phone number
delivery confirmation
updates message status
displays the message
sends "read" confirmation
updates message status for participants
sends "read" status update

Class Diagram

This section outlines the main conceptual classes within the TUASIL system, detailing their key attributes and associated methods derived from the functional requirements and implicit data model.

1. Class User

Attributes:

- `userID`: Unique identifier for the user (Maps to `users.id`).
- `phoneNumber`: The user's valid phone number used for registration and login (Maps to `users.phone_number`).
- `username`: Unique public username for the user (Maps to `users.username`).
- `firstName`: User's first name (Maps to `users.first_name`).
- `lastName`: User's last name (Maps to `users.last_name`).
- `bio`: User's personal bio/description (Maps to `users.bio`).
- `profilePictureURL`: URL to the user's profile picture (Maps to `users.profile_picture_url`).
- `isOnline`: Boolean indicating user's current online status (Maps to `users.is_online`).
- `lastSeenAt`: Timestamp of the user's last activity (Maps to `users.last_seen_at`).
- `privacySettings`: Reference to associated `UserSettings` object, defining privacy preferences.

Methods:

- `register(phoneNumber)`: Initiates user registration.
- `authenticate(otp)`: Authenticates the user with an OTP.
- `updateProfile(data)`: Updates user's username, name, bio, or profile picture.
- `viewOnlineStatus(contact)`: Checks and displays the online status of a contact (respecting privacy).
- `viewLastSeen(contact)`: Checks and displays the last seen status of a contact (respecting privacy).
- `configurePrivacy(setting, value)`: Updates specific privacy settings.

2. Class Chat

Attributes:

- `chatID`: Unique identifier for the chat (Maps to `chats.id`).
- `chatType`: Type of chat (e.g., 'private', 'group', 'channel') (Maps to `chats.chat_type`).
- `chatName`: Name of the chat (for groups/channels) (Maps to `chats.chat_name`).
- `chatPictureURL`: URL to the chat's picture (Maps to `chats.chat_picture_url`).
- `chatDescription`: Description for the chat (for groups/channels) (Maps to `chats.chat_description`).
- `publicLink`: Public link for channels (Maps to `chats.public_link`).
- `createdAt`: Timestamp of chat creation.

Methods:

- `createPrivateChat(otherUserID)`: Initiates a one-on-one chat.
- `createGroupChat(name, initialMembers)`: Creates a new group chat.
- `createChannel(name, description, isPublic)`: Creates a new channel.
- `updateChatDetails(name, picture, description)`: Updates chat information (for admins).
- `deleteChat(forUserOnly)`: Deletes the chat for the user, or potentially for all.

3. Class Message

Attributes:

- `messageID`: Unique identifier for the message (Maps to `messages.id`).
- `chatID`: ID of the chat the message belongs to (Maps to `messages.chat_id`).
- `senderID`: ID of the user who sent the message (Maps to `messages.sender_id`).
- `content`: The text content of the message (Maps to `messages.content`).
- `messageType`: Type of message (e.g., 'text', 'image', 'video', 'file') (Maps to `messages.message_type`).
- `sentAt`: Timestamp when the message was sent (Maps to `messages.sent_at`).
- `mediaID`: ID of associated media, if any (Maps to `messages.media_id`).
- `repliedToMessageID`: ID of the message this message is a reply to (Maps to `messages.replied_to_message_id`).
- `forwardedFromUserID`: ID of the original sender if forwarded (Maps to `messages.forwarded_from_user_id`).
- `forwardedFromChatID`: ID of the original chat if forwarded (Maps to `messages.forwarded_from_chat_id`).
- `editedAt`: Timestamp if the message was edited (Maps to `messages.edited_at`).
- `isDeleted`: Boolean flag for soft deletion (Maps to `messages.is_deleted`).
- `viewCount`: Number of views for channel messages (Maps to `messages.view_count`).
- `status`: Current status (e.g., 'sent', 'delivered', 'read').

Methods:

- `send(chat, sender, content, type, mediaID)`: Sends a new message.
- `receive()`: Processes an incoming message.
- `edit(newContent)`: Edits the message content (if allowed).
- `delete(forAll)`: Deletes the message, optionally for all participants.
- `reply(parentMessageID)`: Creates a reply to a specific message.
- `forward(targetChatID)`: Forwards the message to another chat.
- `updateStatus(newStatus)`: Updates the delivery/read status.

4. Class Media

Attributes:

- `mediaID`: Unique identifier for the media file (Maps to `media.id`).
- `messageID`: ID of the message this media is attached to (Maps to `media.message_id`).
- `filePathOrURL`: Storage path or URL of the media file (Maps to `media.file_path_or_url`).
- `thumbnailURL`: URL to the thumbnail (Maps to `media.thumbnail_url`).
- `fileSize`: Size of the media file in bytes (Maps to `media.file_size`).
- `mediaType`: Type of media (e.g., 'image', 'video', 'document') (Maps to `media.media_type`).
- `uploadedAt`: Timestamp when the media was uploaded.

Methods:

- `upload(file)`: Uploads a media file.
- `retrieve()`: Retrieves the media file.
- `generateThumbnail()`: Generates a thumbnail for the media.

5. Class ChatParticipant

Attributes:

- `participantID`: Unique identifier for the participant entry.
- `userID`: ID of the user participating in the chat (Maps to `chat_participants.user_id`).
- `chatID`: ID of the chat the user is participating in (Maps to `chat_participants.chat_id`).
- `role`: User's role in the chat (e.g., 'member', 'admin', 'creator', 'subscriber') (Maps to `chat_participants.role`).
- `unreadCount`: Number of unread messages for this user in this chat (Maps to `chat_participants.unread_count`).
- `lastReadMessageID`: ID of the last message read by this user in this chat (Maps to `chat_participants.last_read_message_id`).
- `joinedAt`: Timestamp when the user joined the chat.

Methods:

- `addMember(userID)`: Adds a user to a group/channel (for admins).
- `removeMember(userID)`: Removes a user from a group/channel (for admins).
- `updateRole(userID, newRole)`: Changes a member's role (for admins).
- `leaveChat()`: Allows a user to leave a group chat.
- `resetUnreadCount()`: Resets the unread message count when chat is viewed.

6. Class Contact

Attributes:

- `contactID`: Unique identifier for the contact entry.
- `userID`: ID of the user who owns this contact list (Maps to `contacts.user_id`).
- `contactUserID`: ID of the user being added as a contact (Maps to `contacts.contact_user_id`).
- `aliasName`: Custom alias name for the contact (Maps to `contacts.alias_name`).
- `createdAt`: Timestamp when the contact was added.

Methods:

- `addContact(targetUserID)`: Adds another user as a contact.
- `viewContacts()`: Retrieves the list of contacts.
- `assignAlias(contactUserID, alias)`: Assigns an alias name to a contact.
- `blockContact(contactUserID)`: Blocks a contact (creates a `BlockedUser` entry).

7. Class Notification

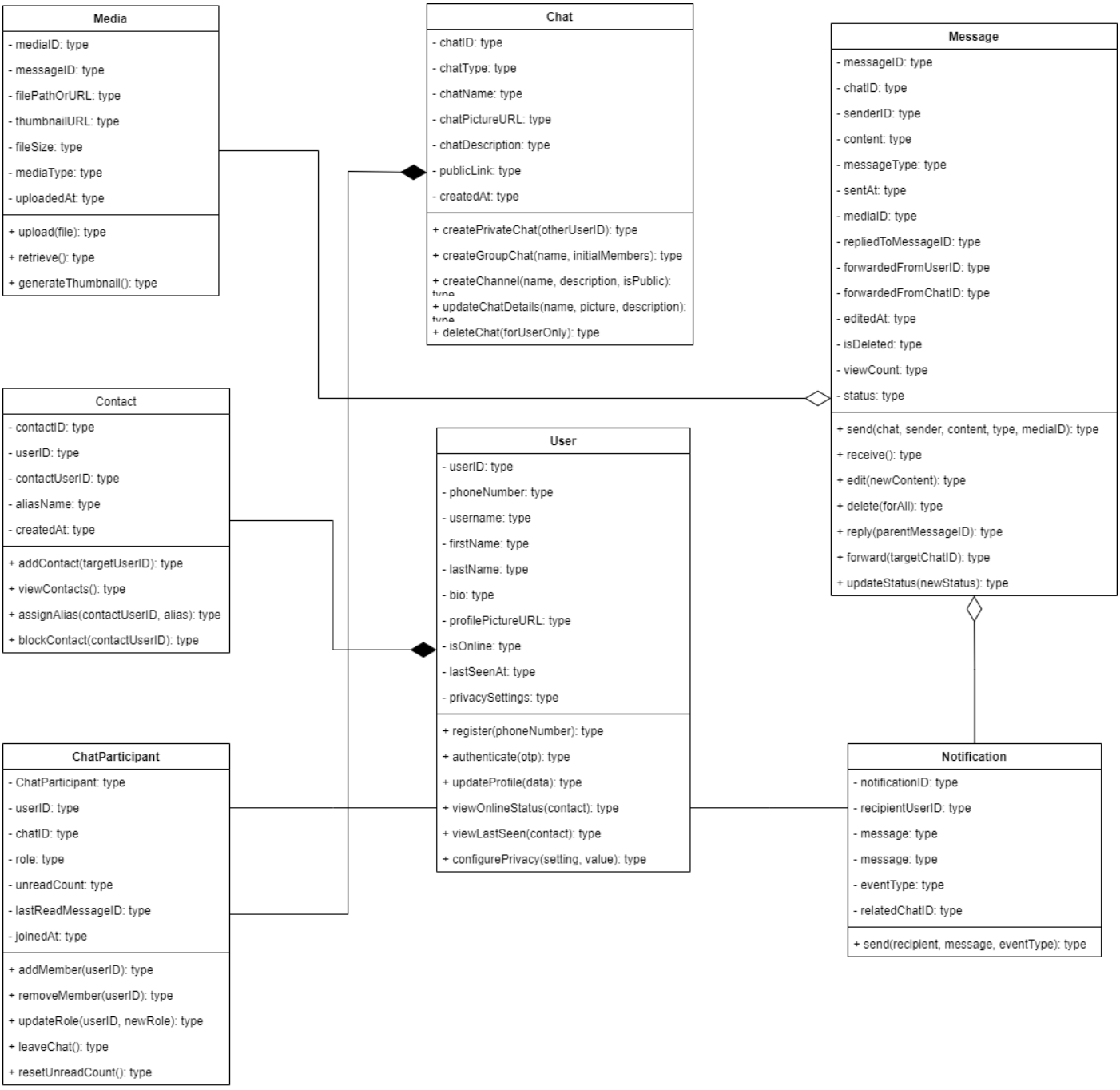
Attributes:

- `notificationID`: Unique identifier for the notification.
- `recipientUserID`: The ID of the user receiving the notification.
- `message`: The content of the notification (e.g., "New message from X").
- `timestamp`: The time the notification was generated/sent.
- `eventType`: The type of event triggering the notification (e.g., 'new_message', 'group_add').
- `relatedChatID`: ID of the chat related to the notification.

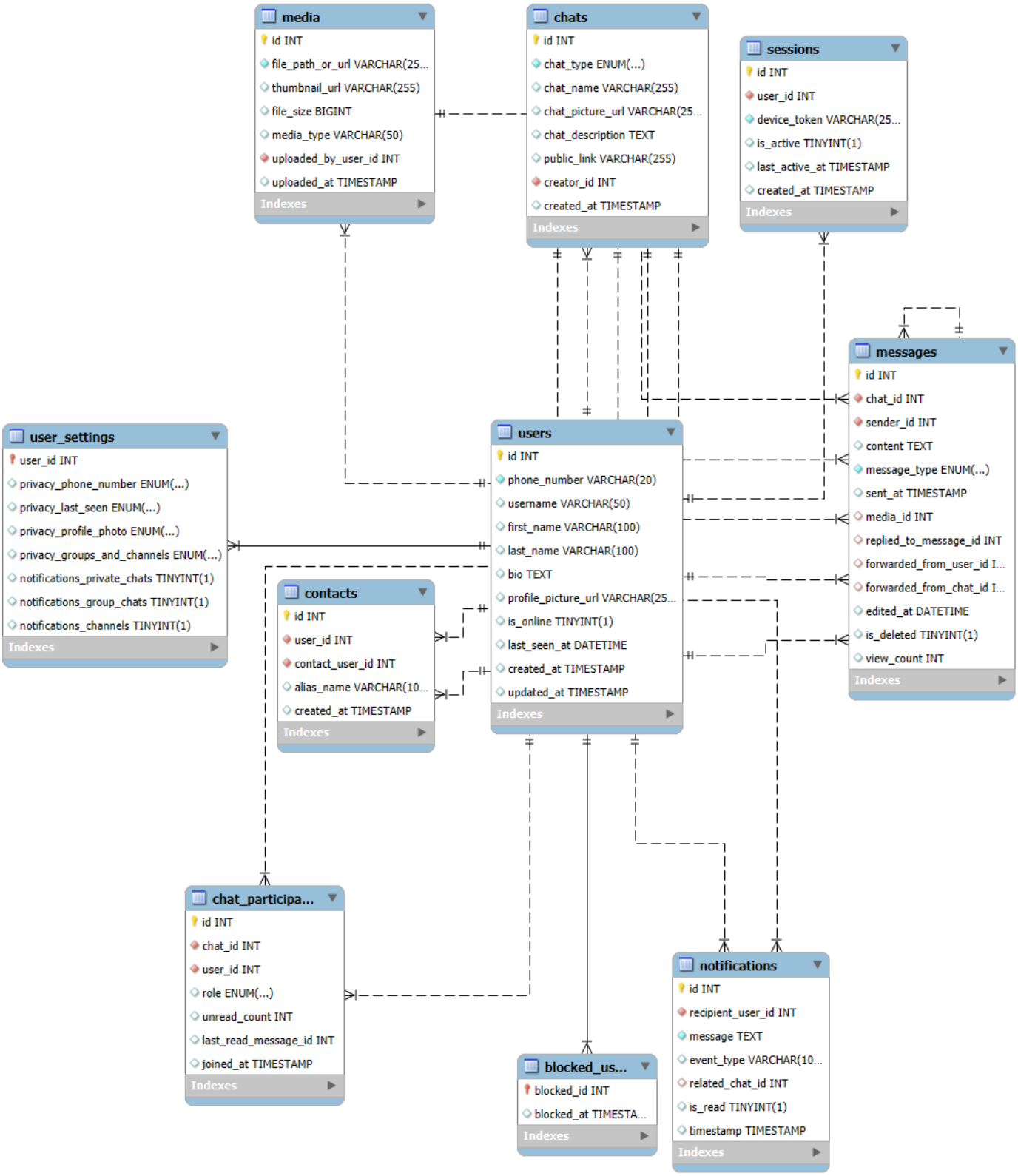
Methods:

- `send(recipient, message, eventType)`: Sends a push notification to a user's active devices.

Class diagram



Database EERD



data flow diagram (context Diagram)

1. Central Process: TUASIL - Secure Messaging Platform

- This is the core of the diagram, representing the entire messaging application, including its client-side logic, backend servers, and internal data handling.

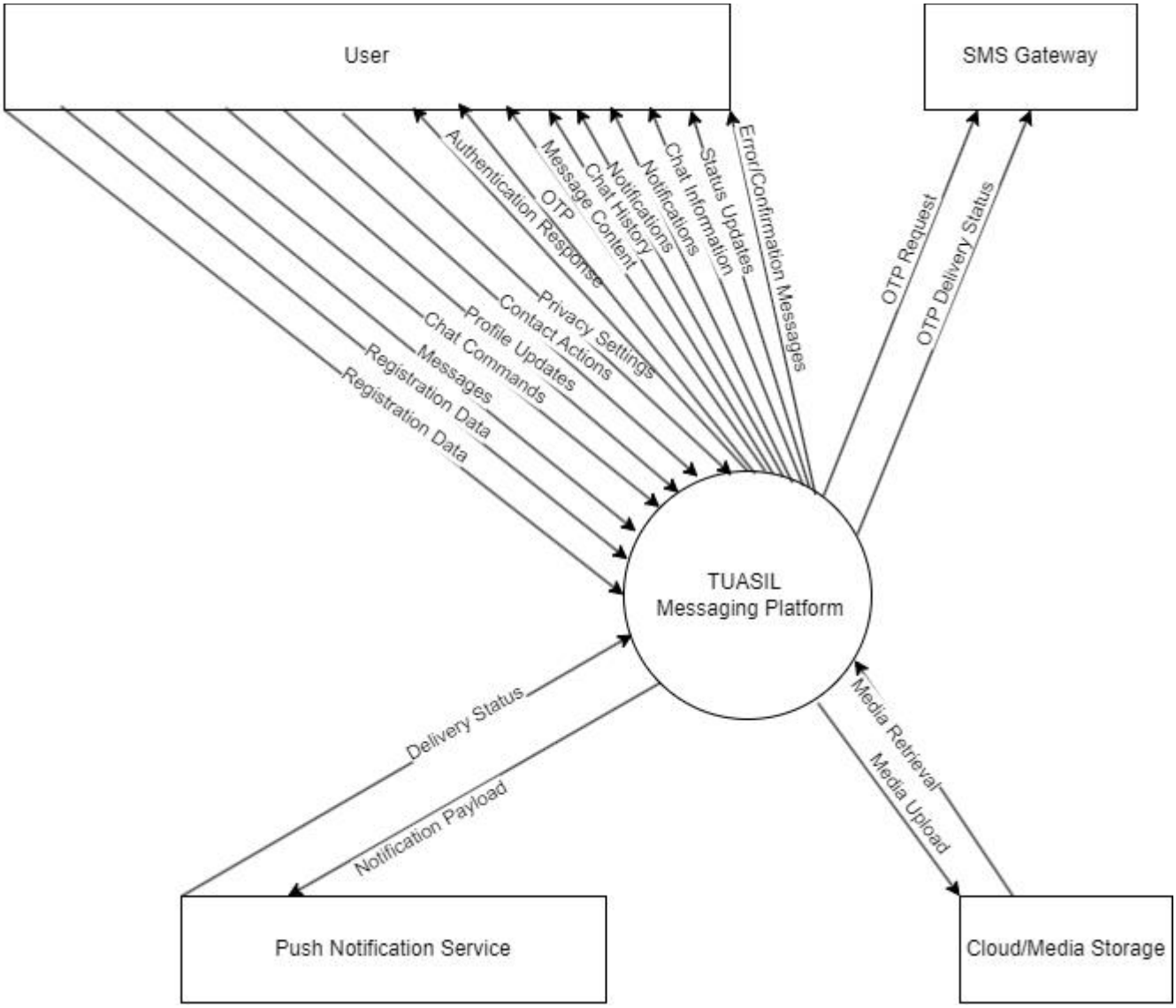
2. External Entities:

- **User:**
  - This is the primary human actor. It represents any individual who uses the TUASIL mobile application to communicate.
- **SMS Gateway:**
  - An external third-party service responsible for sending SMS messages, specifically for delivering One-Time Passwords (OTPs) during user registration and authentication.
- **Push Notification Service (e.g., APNS/FCM):**
  - An external third-party service (like Apple Push Notification Service or Firebase Cloud Messaging) that facilitates the delivery of real-time notifications to users' mobile devices, even when the app is closed.
- **Cloud/Media Storage Service:**
  - An external service or dedicated storage infrastructure used by the TUASIL system to store large media files (images, videos, documents) securely and scalably.

3. Data Flows:

- **Between User and TUASIL Platform:**
  - **User Input / Requests (to System):**
    - **Registration Data:** Phone number, desired username, profile details.
    - **Authentication Data:** OTPs, login credentials.
    - **Messages (Text/Media):** Content of messages, media files for upload.
    - **Chat Commands:** Create/join/leave chats, add/remove members, manage roles, delete messages.
    - **Profile Updates:** Changes to name, bio, profile picture.
    - **Contact Actions:** Add contact, block user, assign alias.
    - **Privacy Settings:** User preferences for visibility and group invitations.
  - **System Responses / Data (to User):**
    - **Authentication Response:** Success/failure, authentication tokens.
    - **OTP:** Sent during registration/login.
    - **Message Content:** Incoming text messages, media for download/display.
    - **Chat History:** Past messages when a chat is opened.
    - **Notifications:** In-app alerts, status updates (e.g., message delivered/read).
    - **Profile Data:** User's own profile information, contact profiles.
    - **Chat Information:** Group/channel details, participant lists.
    - **Status Updates:** Online status, typing indicators.
    - **Error/Confirmation Messages:** System feedback.
- **Between SMS Gateway and TUASIL Platform:**
  - **OTP Request (to SMS Gateway):** TUASIL sends requests to the SMS Gateway to deliver OTPs to users.
  - **OTP Delivery Status (from SMS Gateway):** The SMS Gateway confirms successful or failed OTP delivery.
- **Between Push Notification Service and TUASIL Platform:**
  - **Notification Payload (to Push Notification Service):** TUASIL sends notification content and recipient device tokens to the Push Notification Service.

    - **Delivery Status (from Push Notification Service):** The Push Notification Service provides feedback on notification delivery (less critical for a high-level context diagram, often omitted or implied).
- **Between Cloud/Media Storage Service and TUASIL Platform:**
    - **Media Upload (to Cloud Storage):** TUASIL sends media files from users to the storage service.
    - **Media Retrieval (from Cloud Storage):** TUASIL retrieves media files from the storage service to deliver to users.



Level 1 DFD

1. Main Processes:

The TUASIL system can be logically divided into the following key functional sub-processes:

- **P1: User & Session Management:** Handles all aspects of user lifecycle, including registration, authentication, profile management, and maintaining user session information for push notifications.
- **P2: Chat & Channel Management:** Manages the creation, modification, and deletion of different chat types (private, group, channel) and their participants.
- **P3: Messaging & Media Handling:** Governs the sending, receiving, editing, deleting, and storing of messages and associated media files.
- **P4: Contact & Blocking Management:** Deals with managing user contacts and implementing blocking functionalities.
- **P5: Notification & Real-time Status Management:** Responsible for delivering notifications and managing real-time user statuses like online presence and typing indicators.

2. Data Stores:

The system relies on several internal data stores to persist information:

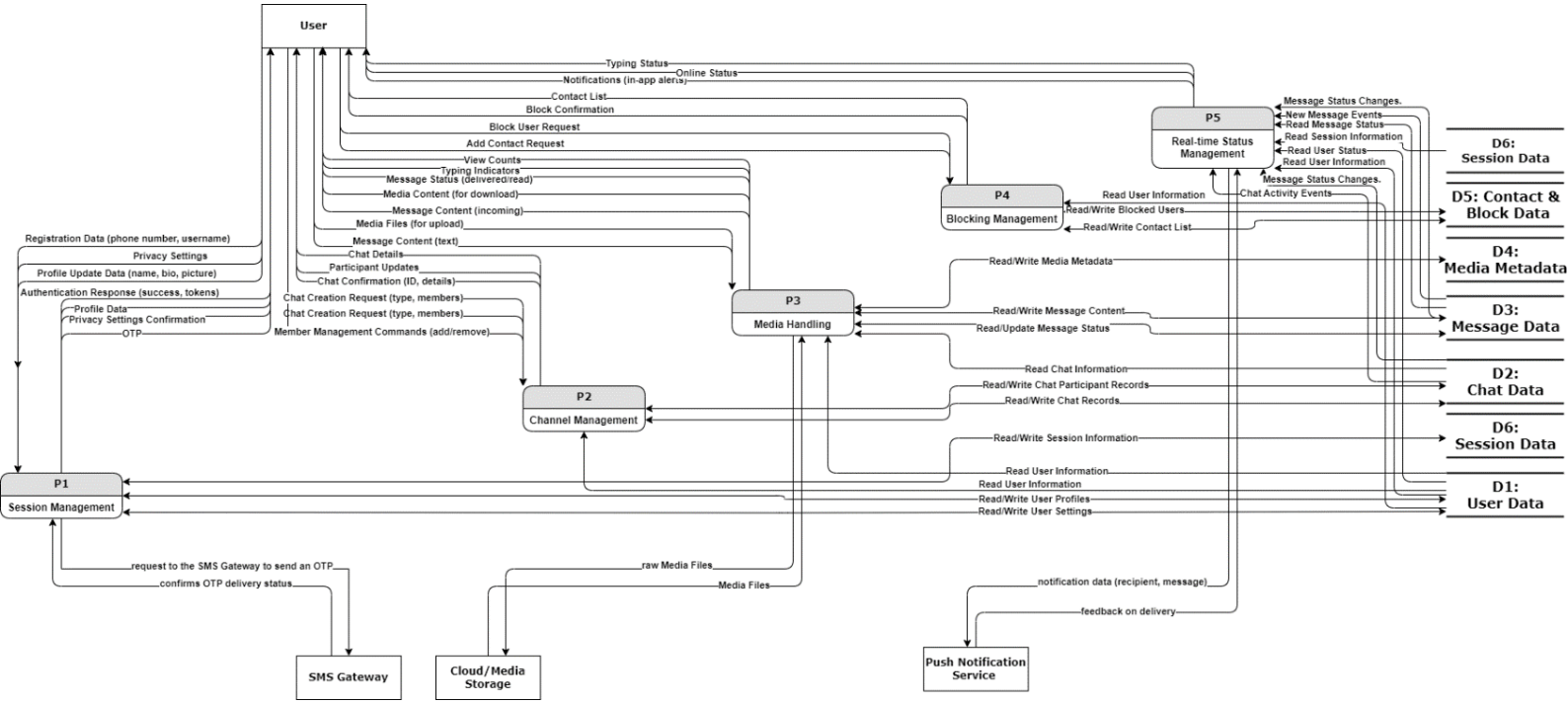
- **D1: User Data:** Stores user profiles, authentication credentials (e.g., hashed passwords, OTP verification status), and user-specific settings (privacy, notifications).
- **D2: Chat Data:** Contains information about chats (groups, channels, private chat metadata) and participants within each chat (roles, unread counts).
- **D3: Message Data:** Stores all messages, including their content, sender, timestamp, and metadata like replies, forwards, and status.
- **D4: Media Metadata:** Stores metadata about uploaded media files, such as URLs, file sizes, and types (actual large files are in external storage).
- **D5: Contact & Block Data:** Holds records of user contacts and explicitly blocked user relationships.
- **D6: Session Data:** Manages active user sessions and device tokens necessary for push notifications.

3. Data Flows:

Below are the primary data flows between the processes, external entities, and data stores:

- **Flows Involving 'User' (External Entity):**
    - **User Input / Requests (to P1):** User provides **Registration Data** (phone number, username), **Authentication Data** (OTP), **Profile Update Data** (name, bio, picture), and **Privacy Settings**.
    - **User Input / Requests (to P2):** User sends **Chat Creation Request** (type, members), **Member Management Commands** (add/remove), **Join/Leave Chat Requests**, **Chat Details Update**.
    - **User Input / Requests (to P3):** User sends **Message Content** (text), **Media Files** (for upload).
    - **User Input / Requests (to P4):** User sends **Add Contact Request**, **Block User Request**.
    - **System Responses / Data (from P1): Authentication Response** (success, tokens), **Profile Data**, **Privacy Settings Confirmation**, **OTP**.
    - **System Responses / Data (from P2): Chat Confirmation** (ID, details), **Participant Updates**, **Chat Details**.
    - **System Responses / Data (from P3): Message Content** (incoming), **Media Content** (for download), **Message Status** (delivered/read), **Typing Indicators**, **View Counts**.

    - **System Responses / Data (from P4): Contact List**, **Block Confirmation**.
    - **System Responses / Data (from P5): Notifications** (in-app alerts), **Online Status**, **Typing Status**.
- **Flows Involving 'SMS Gateway' (External Entity):**
    - **OTP Request (from P1):** P1 sends a request to the SMS Gateway to send an OTP.
    - **OTP Delivery Status (to P1):** SMS Gateway confirms OTP delivery status back to P1.
- **Flows Involving 'Push Notification Service' (External Entity):**
    - **Notification Payload (from P5):** P5 sends notification data (recipient, message) to the Push Notification Service.
    - **(Implicit) Notification Delivery Confirmation (to P5):** Push Notification Service provides feedback on delivery (though often internal to the service and not explicitly shown in this level of DFD).
- **Flows Involving 'Cloud/Media Storage Service' (External Entity):**
    - **Media Upload (from P3):** P3 sends raw **Media Files** to the Cloud/Media Storage Service for storage.
    - **Media Retrieval (to P3):** P3 requests **Media Files** from the Cloud/Media Storage Service for delivery to users.
- **Flows Between Processes and Data Stores:**
    - **P1: User & Session Management <-> D1: User Data: Read/Write User Profiles**, **Read/Write User Settings**.
    - **P1: User & Session Management <-> D6: Session Data: Write Session Information** (on login), **Read Session Information** (on logout, or by P5 for notifications).
    - **P2: Chat & Channel Management <-> D2: Chat Data: Read/Write Chat Records**, **Read/Write Chat Participant Records**.
    - **P2: Chat & Channel Management -> D1: User Data: Read User Information** (for member details).
    - **P3: Messaging & Media Handling <-> D3: Message Data: Read/Write Message Content**, **Read/Update Message Status**.
    - **P3: Messaging & Media Handling <-> D4: Media Metadata: Read/Write Media Metadata** (e.g., URL after upload).
    - **P3: Messaging & Media Handling -> D2: Chat Data: Read Chat Information** (for message context).
    - **P3: Messaging & Media Handling -> D1: User Data: Read User Information** (for sender/recipient details).
    - **P4: Contact & Blocking Management <-> D5: Contact & Block Data: Read/Write Contact List**, **Read/Write Blocked Users**.
    - **P4: Contact & Blocking Management -> D1: User Data: Read User Information** (for contact details).
    - **P5: Notification & Real-time Status Management -> D1: User Data: Read User Status** (for online/last seen).
    - **P5: Notification & Real-time Status Management -> D6: Session Data: Read Session Information** (to find active devices for notifications).
    - **P5: Notification & Real-time Status Management -> D3: Message Data: Read Message Status** (for unread counts).
    - **P5: Notification & Real-time Status Management <- P3: Messaging & Media Handling: New Message Events**, **Message Status Changes**.
    - **P5: Notification & Real-time Status Management <- P2: Chat & Channel Management: Chat Activity Events** (e.g., member added/removed).


User
Typing Status
Online Status
Notifications (in-app alerts)
Contact List
Block Confirmation
Block User Request
Add Contact Request
View Counts
Typing Indicators
Message Status (delivered/read)
Media Content (for download)
Message Content (incoming)
Media Files (for upload)
Message Content (text)
Chat Details
Participant Updates
Chat Confirmation (ID, details)
Chat Creation Request (type, members)
Chat Creation Request (type, members)
Member Management Commands (add/remove)
Registration Data (phone number, username)
Privacy Settings
Profile Update Data (name, bio, picture)
Authentication Response (success, tokens)
Profile Data
Privacy Settings Confirmation
OTP
P1
Session Management
P2
Channel Management
P3
Media Handling
P4
Blocking Management
P5
Real-time Status Management
Message Status Changes.
New Message Events
Read Message Status
Read Session Information
Read User Status
Read User Information
Message Status Changes.
Chat Activity Events
Read User Information
Read/Write Blocked Users
Read/Write Contact List
Read/Write Media Metadata
Read/Write Message Content
Read/Update Message Status
Read Chat Information
Read/Write Chat Participant Records
Read/Write Chat Records
Read/Write Session Information
Read User Information
Read User Information
Read/Write User Profiles
Read/Write User Settings
D6: Session Data
D5: Contact & Block Data
D4: Media Metadata
D3: Message Data
D2: Chat Data
D6: Session Data
D1: User Data
request to the SMS Gateway to send an OTP
confirms OTP delivery status
raw Media Files
Media Files
notification data (recipient, message)
feedback on delivery
SMS Gateway
Cloud/Media Storage
Push Notification Service

Level 2 DFD

Level 2 Data Flow Diagram: Detailed Flows for User & Session Management

1. Sub-Processes:

The "User & Session Management" process is further decomposed into the following sub-processes:

- **P1.1: User Registration:** Manages the entire new user sign-up flow, including phone number validation and initial profile creation.
- **P1.2: User Authentication:** Handles user login attempts, verifies One-Time Passwords (OTPs), and authenticates users into the system, initiating new sessions.
- **P1.3: Profile Management:** Allows authenticated users to view and modify their personal profile information (e.g., username, bio, profile picture).
- **P1.4: Privacy Settings Management:** Provides functionalities for users to define and update their privacy preferences regarding visibility and group invitations.
- **P1.5: Session Management:** Oversees the creation, maintenance, and termination of active user sessions across different devices, crucial for real-time features and push notifications.

2. External Entities (carried over from Context Diagram):

- **User:** The human actor interacting with the system.
- **SMS Gateway:** External service for sending OTPs.

3. Data Stores (relevant to this decomposition):

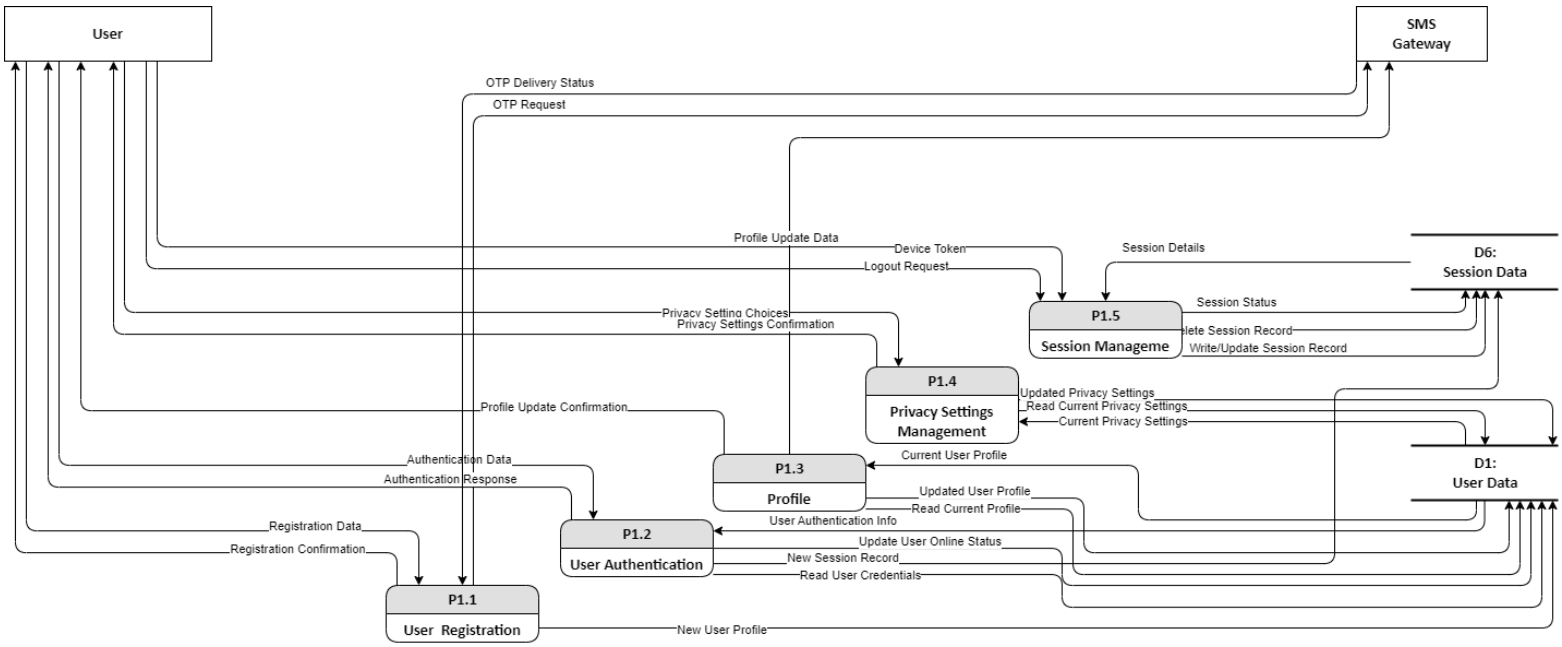
- **D1: User Data:** Stores user profiles, authentication data, and privacy settings.
- **D6: Session Data:** Stores active user sessions and associated device tokens.

4. Data Flows:

Here's a detailed description of the data flows within and around the "User & Session Management" processes:

- **Flows for User Registration (P1.1):**
  - **User -> P1.1:** `Registration Data` (Phone Number).
  - **P1.1 -> SMS Gateway:** `OTP Request` (Phone Number, OTP Code).
  - **SMS Gateway -> P1.1:** `OTP Delivery Status` (Success/Failure).
  - **P1.1 -> D1: User Data:** `New User Profile` (User ID, Phone Number, initial settings).
  - **P1.1 -> User:** `Registration Confirmation` (Success/Failure message).
- **Flows for User Authentication (P1.2):**
  - **User -> P1.2:** `Authentication Data` (Phone Number, OTP).
  - **P1.2 -> D1: User Data:** `Read User Credentials` (Phone Number).
  - **D1: User Data -> P1.2:** `User Authentication Info` (Hashed OTP for verification, User Profile ID).
  - **P1.2 -> D1: User Data:** `Update User Online Status` (User ID, `is_online=TRUE`, `last_seen_at=now`).
  - **P1.2 -> D6: Session Data:** `New Session Record` (User ID, Device Token, Status).
  - **P1.2 -> User:** `Authentication Response` (Authentication Token, User ID, Success/Failure).
- **Flows for Profile Management (P1.3):**
  - **User -> P1.3:** `Profile Update Data` (User ID, New Username, First Name, Last Name, Bio, Profile Picture URL).
  - **P1.3 -> D1: User Data:** `Read Current Profile` (User ID).

    - **D1: User Data -> P1.3:** `Current User Profile` (Existing data for validation).
    - **P1.3 -> D1: User Data:** `Updated User Profile` (User ID, modified fields).
    - **P1.3 -> User:** `Profile Update Confirmation`.
- **Flows for Privacy Settings Management (P1.4):**
    - **User -> P1.4:** `Privacy Setting Choices` (User ID, Setting Name, New Value).
    - **P1.4 -> D1: User Data:** `Read Current Privacy Settings` (User ID).
    - **D1: User Data -> P1.4:** `Current Privacy Settings` (Existing settings).
    - **P1.4 -> D1: User Data:** `Updated Privacy Settings` (User ID, modified preferences).
    - **P1.4 -> User:** `Privacy Settings Confirmation`.
- **Flows for Session Management (P1.5):**
    - **User -> P1.5:** `Device Token` (on app startup/login), `Logout Request` (User ID, Device Token).
    - **P1.5 -> D6: Session Data:** `Write/Update Session Record` (User ID, Device Token, Active Status).
    - **P1.5 -> D6: Session Data:** `Delete Session Record` (on logout).
    - **D6: Session Data -> P1.5:** `Session Details` (for checking active sessions, used by P5).
    - **P1.5 -> User:** `Session Status` (e.g., "Logged out successfully").



Level 2 Data Flow Diagram: Detailed Flows for Chat & Channel Management

1. Sub-Processes:

The "Chat & Channel Management" process is broken down into these core functionalities:

- **P2.1: Chat Creation:** Handles the initiation of all types of chats, including private chats, group chats, and channels, setting their initial properties and participants.
- **P2.2: Participant Management:** Deals with adding, removing, and modifying the roles of members or subscribers within group chats and channels.
- **P2.3: Chat Details Management:** Allows authorized users (typically admins) to update the metadata of a chat, such as its name, picture, and description.
- **P2.4: Chat Joining & Leaving:** Manages user requests to join public channels or to leave group chats, updating their participation status accordingly.

2. External Entities (carried over from Context Diagram):

- **User:** The human actor initiating or affected by chat management actions.

3. Data Stores (relevant to this decomposition):

- **D1: User Data:** Provides user profile information necessary for creating chats or managing participants (e.g., validating user IDs).
- **D2: Chat Data:** The central repository for all chat-related information, including chat details (type, name, public link) and participant records (who is in which chat, with what role, and their read status).

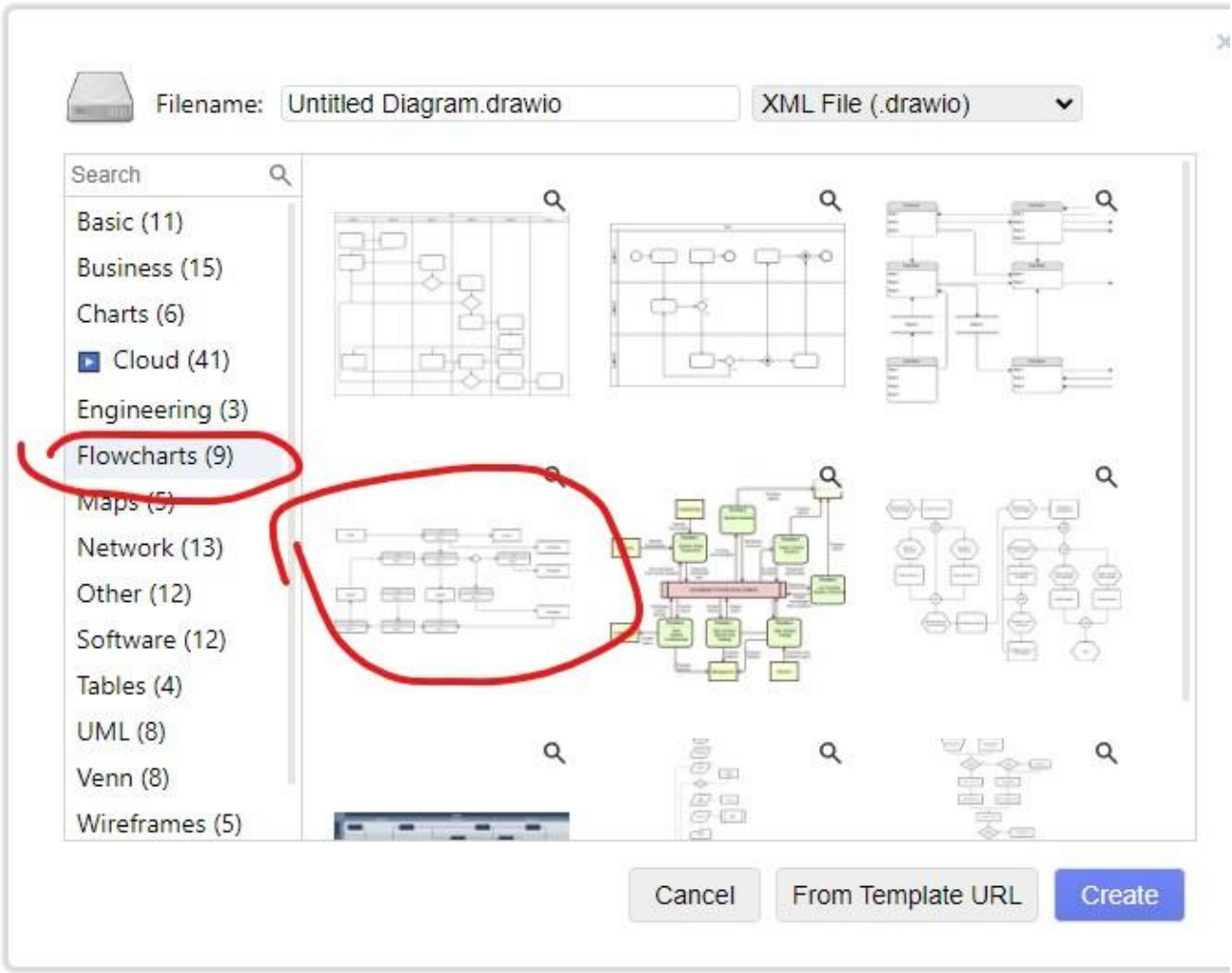
4. Data Flows:

Here are the detailed data flows within and around the "Chat & Channel Management" processes:

- **Flows for Chat Creation (P2.1):**
  - **User -> P2.1:** `Chat Creation Request` (includes Chat Type, desired Name, Description, potential Public Link, and a list of Initial Member IDs).
  - **P2.1 -> D1: User Data:** `Read User Info` (to validate and retrieve details of the creator and initial members).
  - **D1: User Data -> P2.1:** `User Profile Details` (retrieved user information).
  - **P2.1 -> D2: Chat Data:** `New Chat Record` (creates the chat entry with its type, properties, and the creator ID), followed by `Initial Participant Records` (adding the creator and specified members/subscribers with their roles).
  - **D2: Chat Data -> P2.1:** `Chat ID Confirmation` (the unique ID assigned to the newly created chat).
  - **P2.1 -> User:** `Chat Creation Confirmation` (sends back the new Chat ID and details).
- **Flows for Participant Management (P2.2):**
  - **User -> P2.2:** `Member Management Commands` (includes Chat ID, specific User ID to manage, desired Action - e.g., 'Add', 'Remove', 'Update Role', and Admin User ID for authorization).
  - **P2.2 -> D2: Chat Data:** `Read Chat & Participant Info` (to verify admin privileges and current participant list).
  - **D2: Chat Data -> P2.2:** `Current Chat & Participant Details` (for validation).
  - **P2.2 -> D2: Chat Data:** `Updated Participant Records` (adds/removes participants, modifies roles).
  - **P2.2 -> User:** `Participant Status Updates` (e.g., "User added to group", "Role changed").
- **Flows for Chat Details Management (P2.3):**
  - **User -> P2.3:** `Chat Details Update` (includes Chat ID, new Chat Name, Picture URL, Description, and Admin User ID for authorization).
  - **P2.3 -> D2: Chat Data:** `Read Chat Details` (to verify admin privileges for the given chat).
  - **D2: Chat Data -> P2.3:** `Current Chat Details` (for validation).

- **P2.3 -> D2: Chat Data:** `Updated Chat Record` (persists the new name, picture, or description).
- **P2.3 -> User:** `Chat Details Confirmation`.

- **Flows for Chat Joining & Leaving (P2.4):**
  - **User -> P2.4:** `Join/Leave Chat Request` (includes Chat ID, User ID, and for public channels, the Public Link).
  - **P2.4 -> D2: Chat Data:** `Read Chat Info` (to determine chat type, public/private status, and existing participant list).
  - **D2: Chat Data -> P2.4:** `Chat Type & Participant List`.
  - **P2.4 -> D2: Chat Data:** `Updated Participant Record` (adds the user to the chat or removes them).
  - **P2.4 -> User:** `Join/Leave Confirmation`.



Level 2 Data Flow Diagram: Detailed Flows for Messaging & Media Handling

1. Sub-Processes:

The "Messaging & Media Handling" process is decomposed into the following key functionalities:

- **P3.1: Message Sending:** Manages the creation, validation, and initial storage of all outgoing messages, including text and media-related messages.
- **P3.2: Media Upload & Processing:** Handles the secure upload of media files, their processing (e.g., thumbnail generation), and storage.
- **P3.3: Message Delivery:** Orchestrates the real-time or near real-time delivery of messages to recipients' devices.
- **P3.4: Message History & Retrieval:** Manages the storage and retrieval of past messages for display in chat interfaces.
- **P3.5: Message Modification & Deletion:** Handles requests to edit or delete existing messages, updating their status in the system.
- **P3.6: Status & Read Receipt Management:** Manages the tracking and updating of message statuses (sent, delivered, read) and other real-time indicators like typing status.

2. External Entities (carried over from Context Diagram):

- **User:** The human actor sending or receiving messages and media.
- **Cloud/Media Storage Service:** External service for storing large media files.

3. Data Stores (relevant to this decomposition):

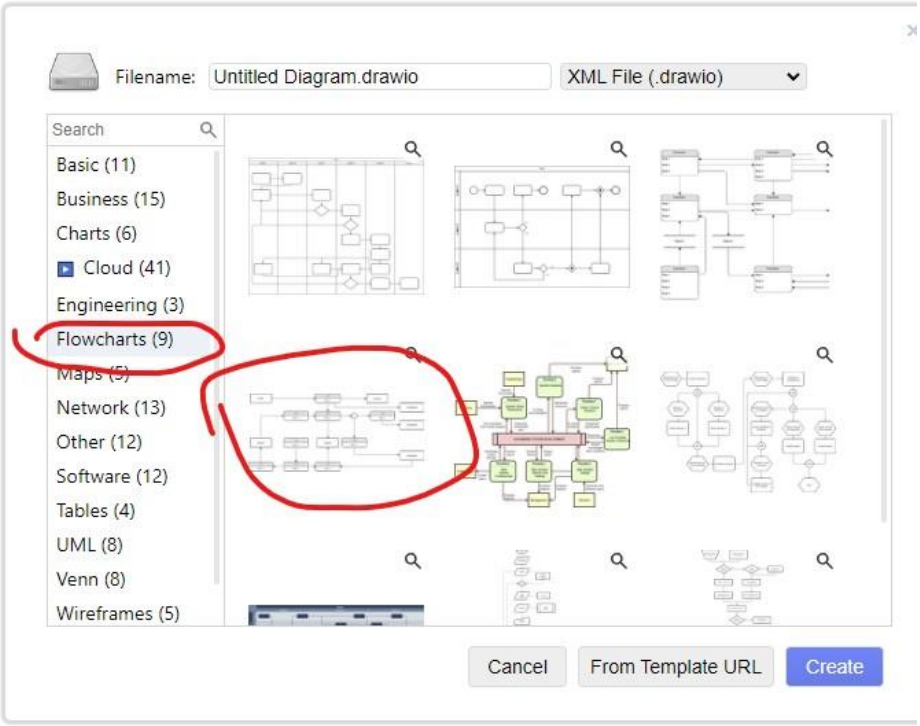
- **D1: User Data:** Provides user profiles and status information (e.g., sender/recipient IDs, online status).
- **D2: Chat Data:** Contains chat metadata and participant information (e.g., to identify message recipients).
- **D3: Message Data:** The primary store for all message content and metadata.
- **D4: Media Metadata:** Stores details about media files (e.g., URLs, types, sizes).
- **D6: Session Data:** Used by P5 for active device information to push notifications (P3.3 would interface with P5 for delivery).

4. Data Flows:

Here are the detailed data flows within and around the "Messaging & Media Handling" processes:

- **Flows for Message Sending (P3.1):**
  - **User -> P3.1:** `Outgoing Message Data` (Chat ID, Sender ID, Content, Message Type, optional Media Attachment).
  - **P3.1 -> D1: User Data:** `Read Sender Profile` (to validate sender).
  - **P3.1 -> D2: Chat Data:** `Read Chat Info` (to validate chat existence and sender's participation).
  - **P3.1 -> D3: Message Data:** `New Message Record` (stores the message with initial 'sent' status).
  - **P3.1 -> P3.3:** `Message for Delivery` (triggers delivery process).
  - **P3.1 -> User:** `Message Sent Confirmation`.
- **Flows for Media Upload & Processing (P3.2):**
  - **User -> P3.2:** `Raw Media File` (Image, Video, Document).
  - **P3.2 -> Cloud/Media Storage Service:** `Media Upload Request` (Raw Media File, Sender ID).
  - **Cloud/Media Storage Service -> P3.2:** `Media Storage Confirmation` (Assigned URL, File Size).
  - **P3.2 -> D4: Media Metadata:** `New Media Metadata` (Media ID, URL, Type, Size, Uploader ID).
  - **P3.2 -> P3.1:** `Media Reference` (Media ID, URL) (for linking to the message).
  - **P3.2 -> User:** `Media Upload Confirmation`.
- **Flows for Message Delivery (P3.3):**
  - **P3.1 -> P3.3:** `Message for Delivery` (Message ID, Chat ID, Sender ID, Recipient IDs).
  - **P3.3 -> D3: Message Data:** `Read Message Content` (retrieves full message details).

    - **P3.3 -> D2: Chat Data:** `Read Chat Participants` (to identify all recipients).
    - **P3.3 -> D6: Session Data:** `Read Active Sessions` (to find recipient devices).
    - **P3.3 -> P5: Notification & Real-time Status Management:** `New Message Notification Request` (Recipient IDs, Message ID, Content Preview).
    - **P3.3 -> User:** `Incoming Message Data` (to active recipient clients).
    - **P3.3 -> D3: Message Data:** `Update Message Status` (Marked as 'delivered' for recipients).
- **Flows for Message History & Retrieval (P3.4):**
    - **User -> P3.4:** `Chat History Request` (Chat ID, User ID, optional Last Message ID/Timestamp).
    - **P3.4 -> D3: Message Data:** `Read Message History` (for specified chat and range).
    - **D3: Message Data -> P3.4:** `Message Records`.
    - **P3.4 -> D4: Media Metadata:** `Read Media Metadata` (for messages with media).
    - **D4: Media Metadata -> P3.4:** `Media Details` (URLs, thumbnails).
    - **P3.4 -> User:** `Chat History Data` (Messages, associated Media URLs).
- **Flows for Message Modification & Deletion (P3.5):**
    - **User -> P3.5:** `Message Modification Request` (Message ID, User ID, New Content for edit, or Delete Flag for deletion).
    - **P3.5 -> D3: Message Data:** `Read Message Details` (to verify sender/admin rights and current status).
    - **D3: Message Data -> P3.5:** `Current Message Record`.
    - **P3.5 -> D3: Message Data:** `Update Message Record` (Set `edited_at`, `is_deleted=TRUE`).
    - **P3.5 -> P3.3:** `Message Update/Delete Notification` (triggers update to recipients).
    - **P3.5 -> User:** `Modification/Deletion Confirmation`.
- **Flows for Status & Read Receipt Management (P3.6):**
    - **User -> P3.6:** `Typing Indicator` (User ID, Chat ID), `Read Confirmation` (Message ID, User ID).
    - **P3.6 -> D3: Message Data:** `Update Message Read Status` (for individual message statuses) / `Update Last Read Message ID` (for participant).
    - **P3.6 -> D2: Chat Data:** `Update Participant Unread Count` (decrements as messages are read).
    - **P3.6 -> P5: Notification & Real-time Status Management:** `Status Update Event` (e.g., Typing status to other chat participants).
    - **P3.6 -> D1: User Data:** `Update User Online Status` (on activity).
    - **P3.6 -> User:** `Read Receipt Display` (to sender).



Level 2 Data Flow Diagram: Detailed Flows for Contact & Blocking Management

1. Sub-Processes:

The "Contact & Blocking Management" process is decomposed into the following functionalities:

- **P4.1: Contact Addition:** Handles the process of a user adding another user to their personal contact list.
- **P4.2: Contact Viewing:** Manages the retrieval and display of a user's entire contact list, including associated profile details.
- **P4.3: Contact Alias Management:** Allows a user to assign a custom, friendly name (alias) to a contact in their list.
- **P4.4: User Blocking:** Facilitates one user blocking another, triggering system-wide restrictions on communication and visibility.

2. External Entities (carried over from Context Diagram):

- **User:** The human actor performing contact management actions.

3. Data Stores (relevant to this decomposition):

- **D1: User Data:** Stores general user profiles, including phone numbers, usernames, and privacy settings, which are essential for identifying and displaying contact information.
- **D5: Contact & Block Data:** The primary repository for contact relationships (who added whom as a contact, with what alias) and records of blocked users.

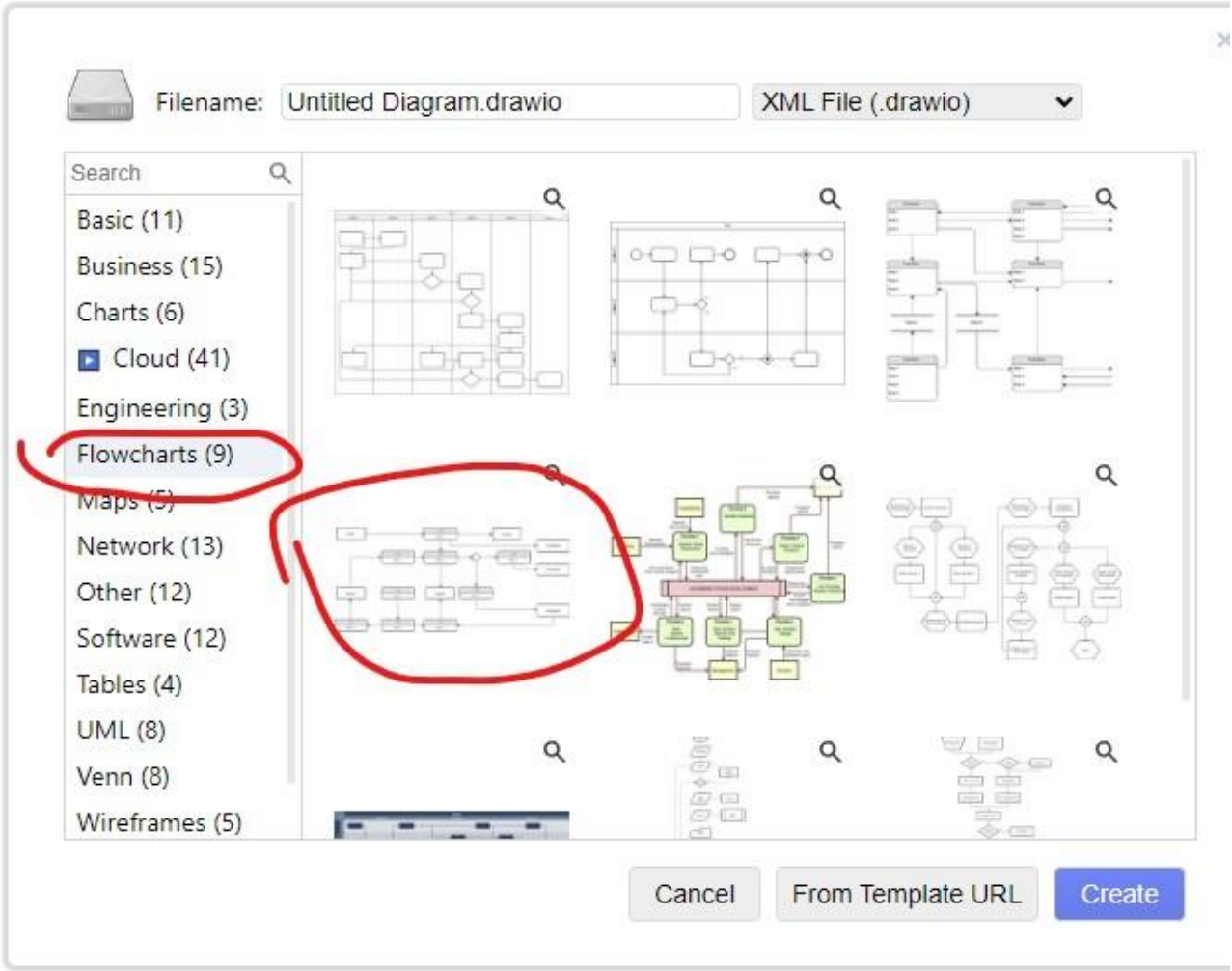
4. Data Flows:

Here are the detailed data flows within and around the "Contact & Blocking Management" processes:

- **Flows for Contact Addition (P4.1):**
  - **User -> P4.1:** `Add Contact Request` (includes User ID of the requester, Phone Number or Username of the target user, and an optional Alias Name).
  - **P4.1 -> D1: User Data:** `Read Target User Info` (P4.1 queries D1 to verify the existence of the target user based on the provided phone number/username).
  - **D1: User Data -> P4.1:** `Target User Profile` (User ID, Public Profile details of the target user if found).
  - **P4.1 -> D5: Contact & Block Data:** `New Contact Record` (persists the relationship: User ID, Target User ID, Alias).
  - **P4.1 -> User:** `Add Contact Confirmation` (Success/Failure message, newly added contact's details).
- **Flows for Contact Viewing (P4.2):**
  - **User -> P4.2:** `View Contacts Request` (includes User ID of the requester).
  - **P4.2 -> D5: Contact & Block Data:** `Read User Contacts` (retrieves all contact entries for the requesting user).
  - **D5: Contact & Block Data -> P4.2:** `Contact IDs List & Aliases` (list of `contact_user_id`s and their `alias_name`s).
  - **P4.2 -> D1: User Data:** `Read Contact Profiles` (P4.2 fetches full profile details for each `contact_user_id`).
  - **D1: User Data -> P4.2:** `Contact Profile Details` (Username, First Name, Last Name, Profile Picture URL, Online Status, Last Seen — respecting privacy settings).
  - **P4.2 -> User:** `Contact List Data` (Formatted list of contacts with their profiles and aliases).
- **Flows for Contact Alias Management (P4.3):**
  - **User -> P4.3:** `Assign Alias Request` (includes User ID, Contact User ID, New Alias Name).

- **P4.3 -> D5: Contact & Block Data:** `Update Contact Alias` (P4.3 modifies the `alias_name` for the specified contact record).
- **D5: Contact & Block Data -> P4.3:** `Contact Record` (read-back for confirmation).
- **P4.3 -> User:** `Alias Update Confirmation`.

- **Flows for User Blocking (P4.4):**
  - **User -> P4.4:** `Block User Request` (includes Blocker User ID, Blocked User ID).
  - **P4.4 -> D1: User Data:** `Read Blocked User Profile` (to confirm the existence of the user to be blocked).
  - **D1: User Data -> P4.4:** `Blocked User Profile` (Confirmation).
  - **P4.4 -> D5: Contact & Block Data:** `New Block Record` (persists the `blocker_id` and `blocked_id` relationship).
  - **P4.4 -> User:** `Block Confirmation` (informs the blocker of success).
  - *(Implicit flow: P4.4 would also trigger internal system events or updates to other processes (e.g., P3: Messaging & Media Handling, P5: Notification & Real-time Status Management) to enforce the blocking restrictions immediately, such as preventing message delivery or hiding status.)*



Level 2 Data Flow Diagram: Detailed Flows for Notification & Real-time Status Management

1. Sub-Processes:

The "Notification & Real-time Status Management" process is broken down into the following key functionalities:

- **P5.1: Event Listener & Parser:** Continuously monitors for various system events (e.g., new messages, user online/offline, typing activity) and parses them into standardized notification or status update requests.
- **P5.2: Notification Dispatcher:** Determines the appropriate recipients and channels for delivering notifications (e.g., active devices, push notification services).
- **P5.3: Status Broadcaster:** Manages the real-time dissemination of user status updates (online/offline, typing) to relevant chat participants.
- **P5.4: Notification Logger:** Records sent notifications for auditing or in-app history.

2. External Entities (carried over from Context Diagram):

- **User:** The human actor who receives notifications or whose status is being managed.
- **Push Notification Service (e.g., APNS/FCM):** An external service responsible for delivering push notifications to mobile devices.

3. Data Stores (relevant to this decomposition):

- **D1: User Data:** Stores user profile information, including their privacy settings (which affect who can see their status).
- **D3: Message Data:** Provides message content and status (e.g., for unread counts, or to build notification previews).
- **D6: Session Data:** Contains records of active user sessions and their associated device tokens, crucial for targeting push notifications.

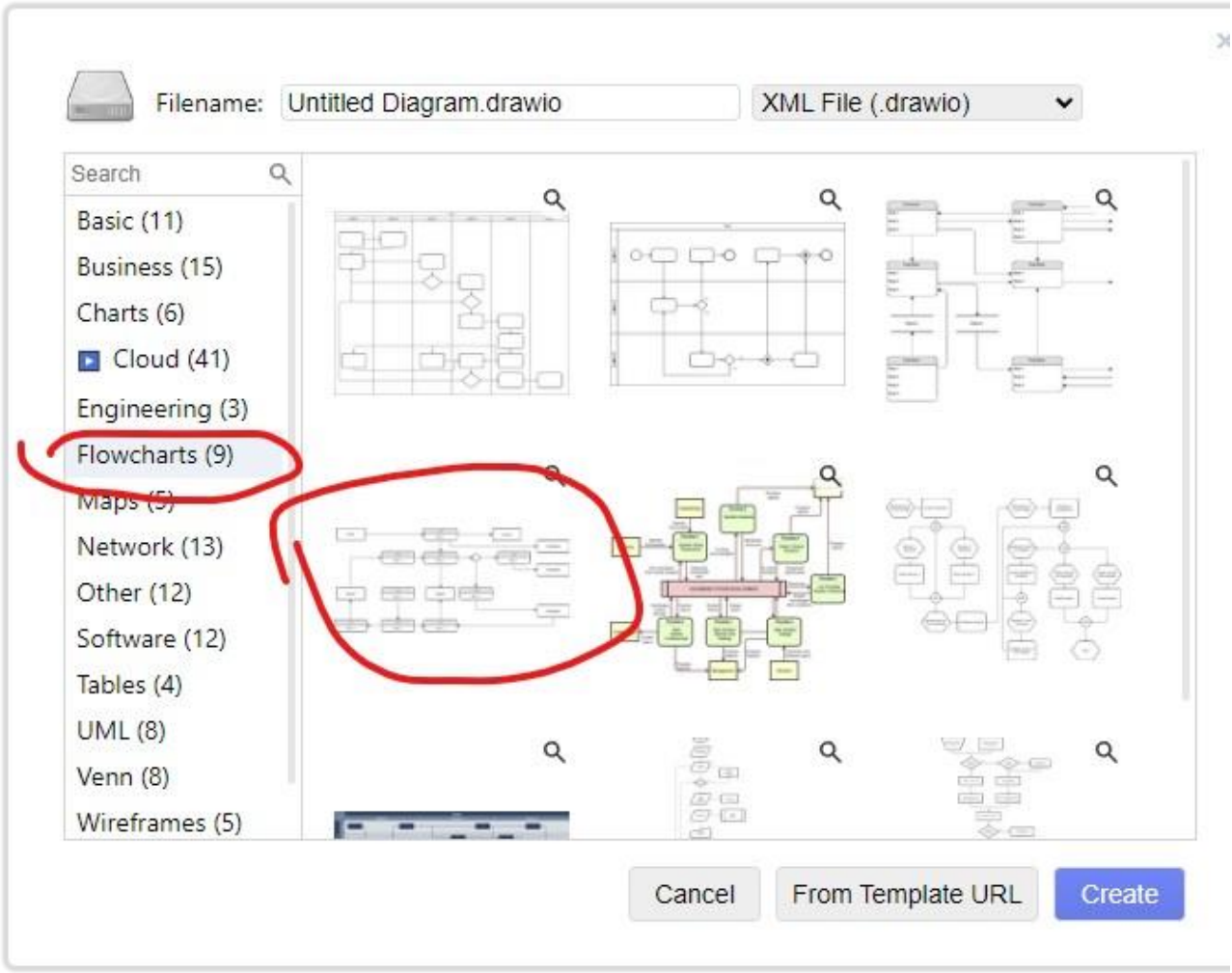
4. Data Flows:

Here are the detailed data flows within and around the "Notification & Real-time Status Management" processes:

- **Flows for Event Listener & Parser (P5.1):**
  - **From P3: Messaging & Media Handling -> P5.1:** `New Message Event` (Message ID, Chat ID, Sender ID, Recipient IDs, Content Preview), `Message Status Change` (Message ID, New Status, Recipient ID).
  - **From P2: Chat & Channel Management -> P5.1:** `Chat Activity Event` (Chat ID, Event Type - e.g., 'Member Added', 'Admin Change').
  - **User -> P5.1:** `Typing Indicator Signal` (User ID, Chat ID, Typing Status).
  - **P5.1 -> P5.2:** `Parsed Notification Request` (Recipient IDs, Notification Type, Content).
  - **P5.1 -> P5.3:** `Status Update Request` (User ID, Status Type - e.g., 'online', 'offline', 'typing', Chat ID).
- **Flows for Notification Dispatcher (P5.2):**
  - **P5.2 -> D6: Session Data:** `Read Active Sessions` (Recipient User IDs).
  - **D6: Session Data -> P5.2:** `Active Device Tokens` (for targeted push notifications).
  - **P5.2 -> D1: User Data:** `Read User Privacy Settings` (to ensure notifications respect user preferences, e.g., for group invites).
  - **D1: User Data -> P5.2:** `User Privacy Preferences`.
  - **P5.2 -> Push Notification Service:** `Notification Payload` (Device Token, Message Text, Notification Type).
  - **P5.2 -> P5.4:** `Notification Log Record` (Details of dispatched notification).
  - **P5.2 -> User:** `In-App Notification` (for active users within the app).
- **Flows for Status Broadcaster (P5.3):**

- **P5.3 -> D1: User Data:** `Read User Online Status` (User ID for whom status is requested).
- **D1: User Data -> P5.3:** `User Online Status`.
- **P5.3 -> D6: Session Data:** `Read Active Sessions` (to identify connected clients for real-time updates).
- **D6: Session Data -> P5.3:** `Active Session Details`.
- **P5.3 -> User:** `Real-time Status Update` (Online/Offline status, Typing indicator) to relevant connected clients.

- **Flows for Notification Logger (P5.4):**
  - **P5.4 -> D3: Message Data:** `Update Unread Counts` (for messages that trigger notifications).
  - **(Optional) P5.4 -> D7: Notification Log (new Data Store):** `Notification Record` (timestamp, recipient, message, status).



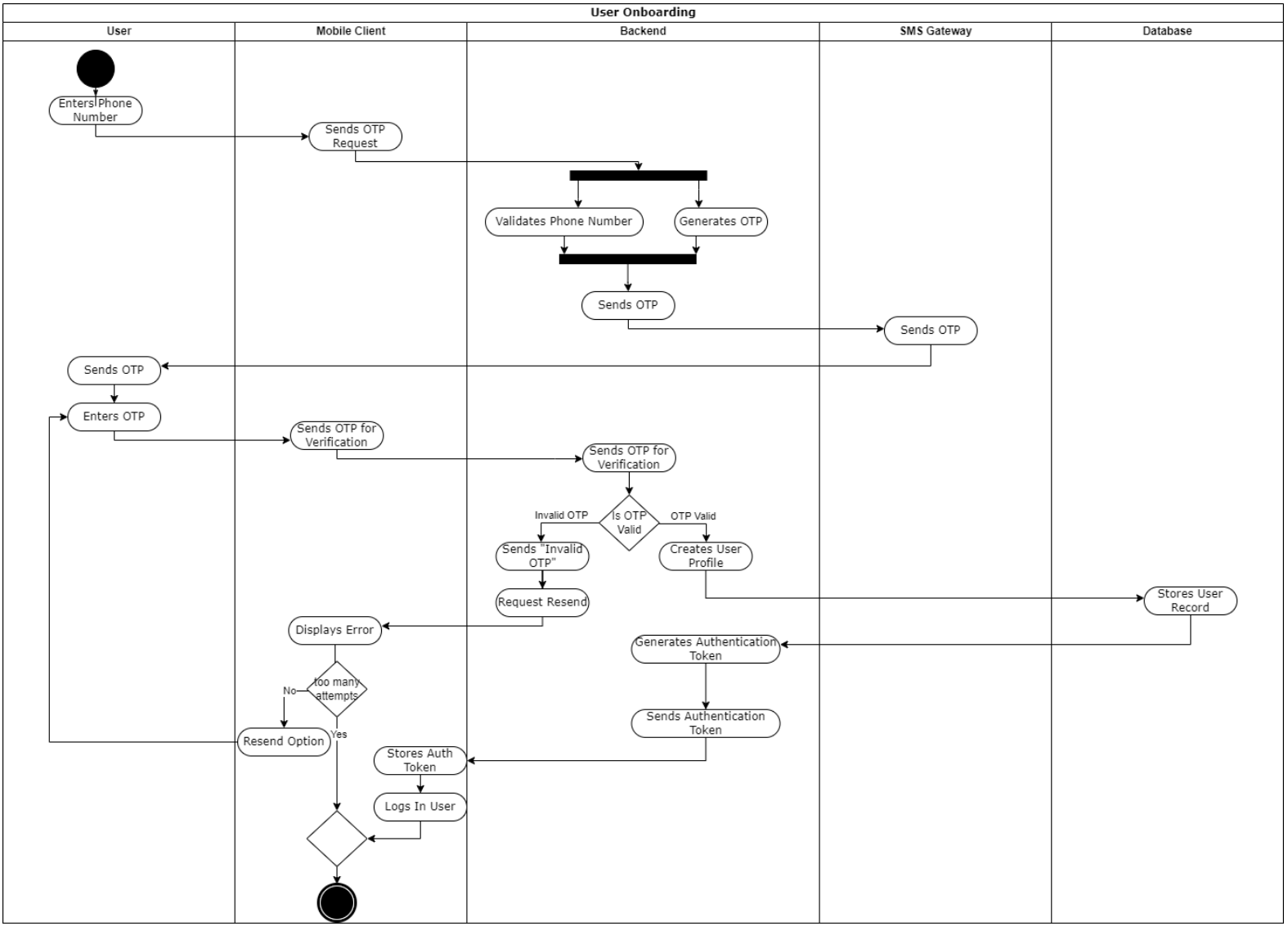
Activity Diagrams

Activity Diagram: User Onboarding and Account Activation

This activity diagram models the complete workflow a user goes through from initiating registration to successfully activating their account and gaining access to the TUASIL platform. It highlights the interactions between the user, their mobile client, the backend system, and the SMS gateway.

Actors/Swimlanes: User, Mobile Client, Backend, SMS Gateway, Database

1. **Start Node:** User wants to join TUASIL.
2. **User (Activity): Enters Phone Number**.
3. **Mobile Client (Activity): Sends OTP Request** (to Backend).
4. **Backend (Activity):**
    - **Validates Phone Number**.
    - **Generates OTP**.
5. **Backend (Activity): Sends OTP via SMS Gateway**.
6. **SMS Gateway (Activity): Delivers OTP** (to User's phone).
7. **User (Activity): Receives OTP** (via SMS).
8. **User (Activity): Enters OTP** (into Mobile Client).
9. **Mobile Client (Activity): Sends OTP for Verification** (to Backend).
10. **Backend (Activity): Verifies OTP**.
11. **Decision Node (Backend): [Is OTP Valid?]**
    - **[No / Invalid OTP / Max Attempts Exceeded]**
        - **Backend (Activity): Sends "Invalid OTP" / "Request Resend"** (to Mobile Client).
        - **Mobile Client (Activity): Displays Error / Resend Option**.
        - *(Flow may loop back to "User enters OTP" or "User requests resend OTP", often with a retry limit.)*
        - **(End Node):** User fails to activate account (e.g., after too many attempts).
    - **[Yes / OTP Valid]**
        - **Backend (Activity): Creates User Profile** (in Database).
        - **Database (Activity): Stores User Record**.
        - **Backend (Activity): Generates Authentication Token**.
        - **Backend (Activity): Sends Authentication Token** (to Mobile Client).
        - **Mobile Client (Activity): Stores Auth Token**.
        - **Mobile Client (Activity): Logs In User**.
        - **End Node:** User account successfully activated and logged in.

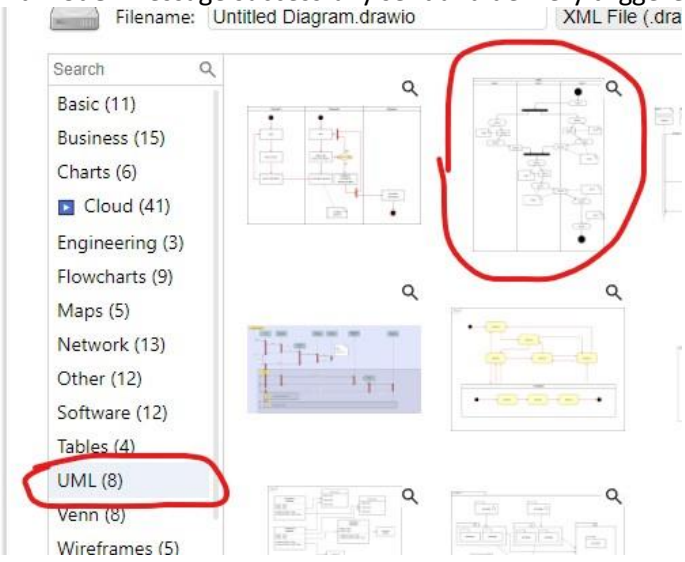

User Onboarding
User
Mobile Client
Backend
SMS Gateway
Database
Enters Phone Number
Sends OTP Request
Validates Phone Number
Generates OTP
Sends OTP
Sends OTP
Sends OTP
Enters OTP
Sends OTP for Verification
Sends OTP for Verification
Is OTP Valid
Invalid OTP
OTP Valid
Sends "Invalid OTP"
Creates User Profile
Request Resend
Stores User Record
Displays Error
Generates Authentication Token
too many attempts
No
Yes
Sends Authentication Token
Resend Option
Stores Auth Token
Logs In User

Activity Diagram: Sending a Message (Text or Media)

This activity diagram illustrates the complete workflow for sending a message in TUASIL, including the conditional path for attaching and uploading media files.

Actors/Swimlanes: User, Mobile Client, Backend (Messaging), Backend (Media), Cloud/Media Storage, Database

1. **Start Node:** User wants to send a message.
2. **User (Activity): Composes Message Content**.
3. **Mobile Client (Activity): Prepares Message Draft**.
4. **Decision Node (Mobile Client): [Attach Media?]**
    - **[No (Text Message)]**
        - **Mobile Client (Activity): Sends Text Message Request** (to Backend - Messaging).
        - *(Continue to common flow at step 10)*
    - **[Yes (Media Message)]**
        - **User (Activity): Selects Media File**.
        - **Mobile Client (Activity): Initiates Media Upload Request** (to Backend - Media).
        - **Backend (Media) (Activity): Requests Pre-signed Upload URL** (from Cloud/Media Storage).
        - **Cloud/Media Storage (Activity): Generates Upload URL**.
        - **Backend (Media) (Activity): Sends Upload URL** (to Mobile Client).
        - **Mobile Client (Activity): Uploads Raw Media File** (directly to Cloud/Media Storage using URL).
        - **Cloud/Media Storage (Activity): Confirms Media Upload Success** (returns permanent Media URL).
        - **Mobile Client (Activity): Sends Media Message Request** (to Backend - Messaging, includes Media URL).
        - *(Continue to common flow at step 10)*
5. **Synchronization Bar (Join Point - Backend):** Both paths (text and media) converge here.
6. **Backend (Messaging) (Activity): Validates Message & Sender**.
7. **Backend (Messaging) (Activity): Stores Message Record** (in Database, including Media ID/URL if applicable).
8. **Database (Activity): Confirms Message Storage**.
9. **Backend (Messaging) (Activity): Triggers Message Delivery** (to Notification/Delivery components).
10. **End Node:** Message successfully sent and delivery triggered.

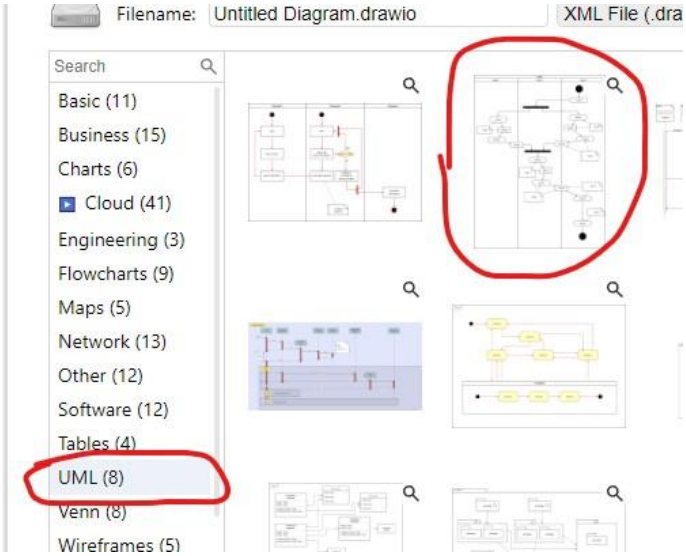


Activity Diagram: Managing Chat Participation

This diagram illustrates the workflow for a user managing participants in a group chat or channel, encompassing actions like joining, leaving, adding new members, or changing existing roles.

Actors/Swimlanes: User, Mobile Client, Backend (Chat Management), Database, Other Mobile Client (for invited users)

1. **Start Node:** User wants to manage chat participation.
2. **User (Activity): Accesses Chat Management Options** (e.g., clicks on group info).
3. **Mobile Client (Activity): Displays Chat Management Menu**.
4. **Decision Node (User Action): [Action Type?]**
    - **[Join Public Channel]**
        - **User (Activity): Inputs Channel Link/ID**.
        - **Mobile Client (Activity): Sends Join Request** (to Backend).
        - **Backend (Chat Management) (Activity): Validates Request & Adds Participant** (in Database).
        - **Database (Activity): Updates Chat Participant Record**.
        - **Backend (Chat Management) (Activity): Sends Join Confirmation** (to Mobile Client).
        - *(Continue to common flow at step 13)*
    - **[Leave Chat]**
        - **User (Activity): Confirms Leave Action**.
        - **Mobile Client (Activity): Sends Leave Request** (to Backend).
        - **Backend (Chat Management) (Activity): Removes Participant** (from Database).
        - **Database (Activity): Updates Chat Participant Record**.
        - **Backend (Chat Management) (Activity): Sends Leave Confirmation** (to Mobile Client).
        - *(Continue to common flow at step 13)*
    - **[Add Member]**
        - **User (Activity): Selects User(s) to Add**.
        - **Mobile Client (Activity): Sends Add Member Request** (to Backend).
        - **Backend (Chat Management) (Activity): Validates Admin Rights & Adds Participants** (in Database).
        - **Database (Activity): Updates Chat Participant Records**.
        - **Backend (Chat Management) (Activity): Triggers New Member Notification** (to Backend - Notification, which then pushes to Other Mobile Clients).
        - *(Continue to common flow at step 13)*
    - **[Change Member Role]**
        - **User (Activity): Selects Member & New Role**.
        - **Mobile Client (Activity): Sends Change Role Request** (to Backend).
        - **Backend (Chat Management) (Activity): Validates Admin Rights & Updates Role** (in Database).
        - **Database (Activity): Updates Participant Role**.
        - **Backend (Chat Management) (Activity): Sends Role Change Confirmation** (to Mobile Client).
        - *(Continue to common flow at step 13)*
5. **Synchronization Bar (Join Point - Backend):** All action paths converge.
6. **Backend (Chat Management) (Activity): Broadcasts Chat Membership Update** (to all relevant Mobile Clients in the chat).
7. **Mobile Client (Activity): Updates Local Chat Roster/UI**.
8. **End Node:** Chat participation action completed.

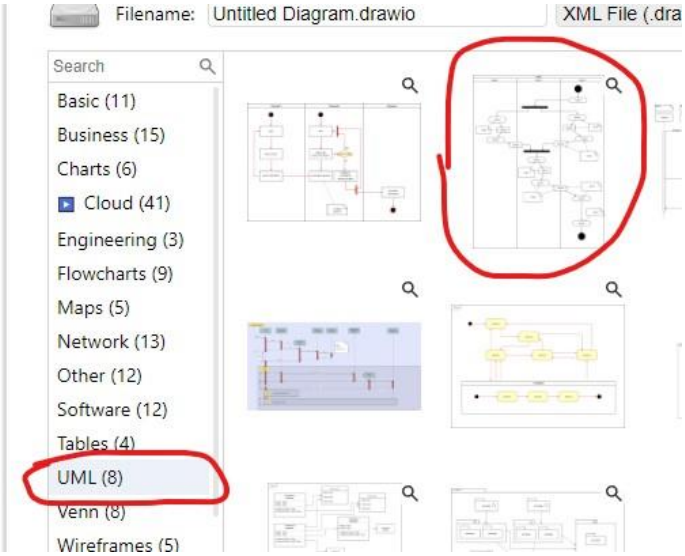


Activity Diagram: User Profile and Privacy Settings Configuration

This diagram outlines the step-by-step process a user follows to access, modify, and save their personal profile information and preferences related to privacy within the TUASIL application.

Actors/Swimlanes: User, Mobile Client, Backend (User Management), Database

1. **Start Node:** User wants to configure settings.
2. **User (Activity): Navigates to Profile/Settings Screen**.
3. **Mobile Client (Activity): Requests Current Profile & Settings Data** (from Backend).
4. **Backend (User Management) (Activity): Retrieves User Profile & Settings** (from Database).
5. **Database (Activity): Returns User Data**.
6. **Backend (User Management) (Activity): Sends User Data** (to Mobile Client).
7. **Mobile Client (Activity): Displays Current Profile & Settings**.
8. **User (Activity): Modifies Profile/Settings Fields**.
9. **User (Activity): Confirms Changes** (e.g., clicks "Save").
10. **Mobile Client (Activity): Sends Update Request** (to Backend, with new data).
11. **Backend (User Management) (Activity): Validates Updated Data**.
12. **Backend (User Management) (Activity): Updates User Profile/Settings** (in Database).
13. **Database (Activity): Confirms Data Update**.
14. **Backend (User Management) (Activity): Sends Update Confirmation** (to Mobile Client).
15. **Mobile Client (Activity): Displays Success Message / Refreshes UI**.
16. **End Node:** Profile/settings successfully updated.



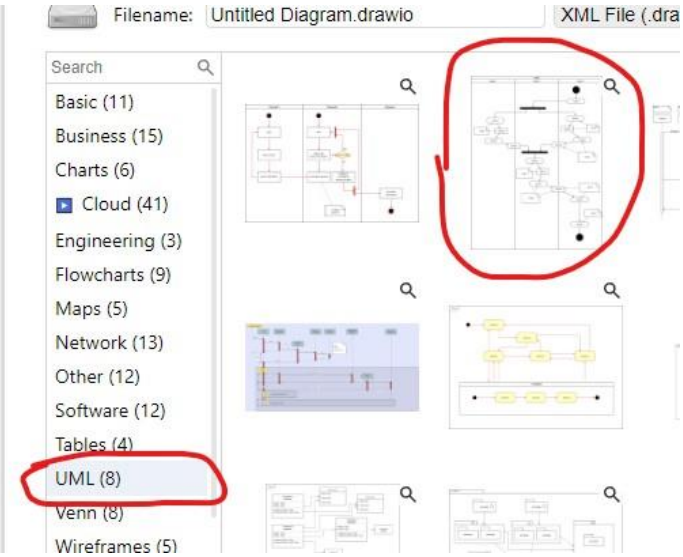
Activity Diagram: End-to-End Chat Interaction

This broader activity diagram illustrates a typical workflow for a user engaged in an ongoing chat conversation, encompassing receiving messages, viewing history, typing, and sending replies.

Actors/Swimlanes: User, Mobile Client, Backend (Messaging), Database, Cloud/Media Storage, Other User (sending messages)

1. **Start Node:** User opens TUASIL app and views chat list.
2. **User (Activity): Selects a Chat**.
3. **Mobile Client (Activity): Requests Chat History** (from Backend - Messaging).
4. **Backend (Messaging) (Activity): Queries Messages & Media Metadata** (from Database).
5. **Database (Activity): Returns Message & Media Metadata**.
6. **Backend (Messaging) (Activity): Sends Chat History Data** (to Mobile Client).
7. **Mobile Client (Activity): Renders Chat History**.
8. **Fork Node (Mobile Client):**
    - **Path A: Receiving New Messages:**

        - **Other User (Activity): Sends New Message**.
        - **Backend (Messaging) (Activity): Pushes New Message** (to Mobile Client).
        - **Mobile Client (Activity): Receives & Displays New Message**.
        - **Mobile Client (Activity): Sends Delivery Confirmation** (to Backend).
        - **Backend (Messaging) (Activity): Updates Message Status to DELIVERED** (in Database).
        - **Mobile Client (Activity): Sends Read Confirmation** (after user views message).
        - **Backend (Messaging) (Activity): Updates Message Status to READ** (in Database).
        - *(Return to Join Node at step 10)*
    - **Path B: User Responds/Interacts:**
        - **User (Activity): Starts Typing**.
        - **Mobile Client (Activity): Sends Typing Indicator** (to Backend).
        - **Backend (Messaging) (Activity): Broadcasts Typing Status** (to other participants).
        - **User (Activity): Composes Message**.
        - **Decision Node (User): [Attach Media?]**
            - **[Yes]**
                - **User (Activity): Selects Media**.
                - **Mobile Client (Activity): Uploads Media** (to Cloud/Media Storage, via Backend - Media).
                - **Cloud/Media Storage (Activity): Confirms Upload & Provides URL**.
                - **Mobile Client (Activity): Sends Media Message Request** (to Backend - Messaging, with URL).
            - **[No]**
                - **Mobile Client (Activity): Sends Text Message Request** (to Backend - Messaging).
        - **Backend (Messaging) (Activity): Stores Message** (in Database).
        - **Backend (Messaging) (Activity): Triggers Message Delivery** (to Notification/Delivery components).
        - *(Return to Join Node at step 10)*
9. **Synchronization Bar (Join Node - Mobile Client):** Both paths (receiving and sending) continue interaction.
10. **End Node:** User exits chat or closes app.



State Diagram

State Diagram: User Account Lifecycle

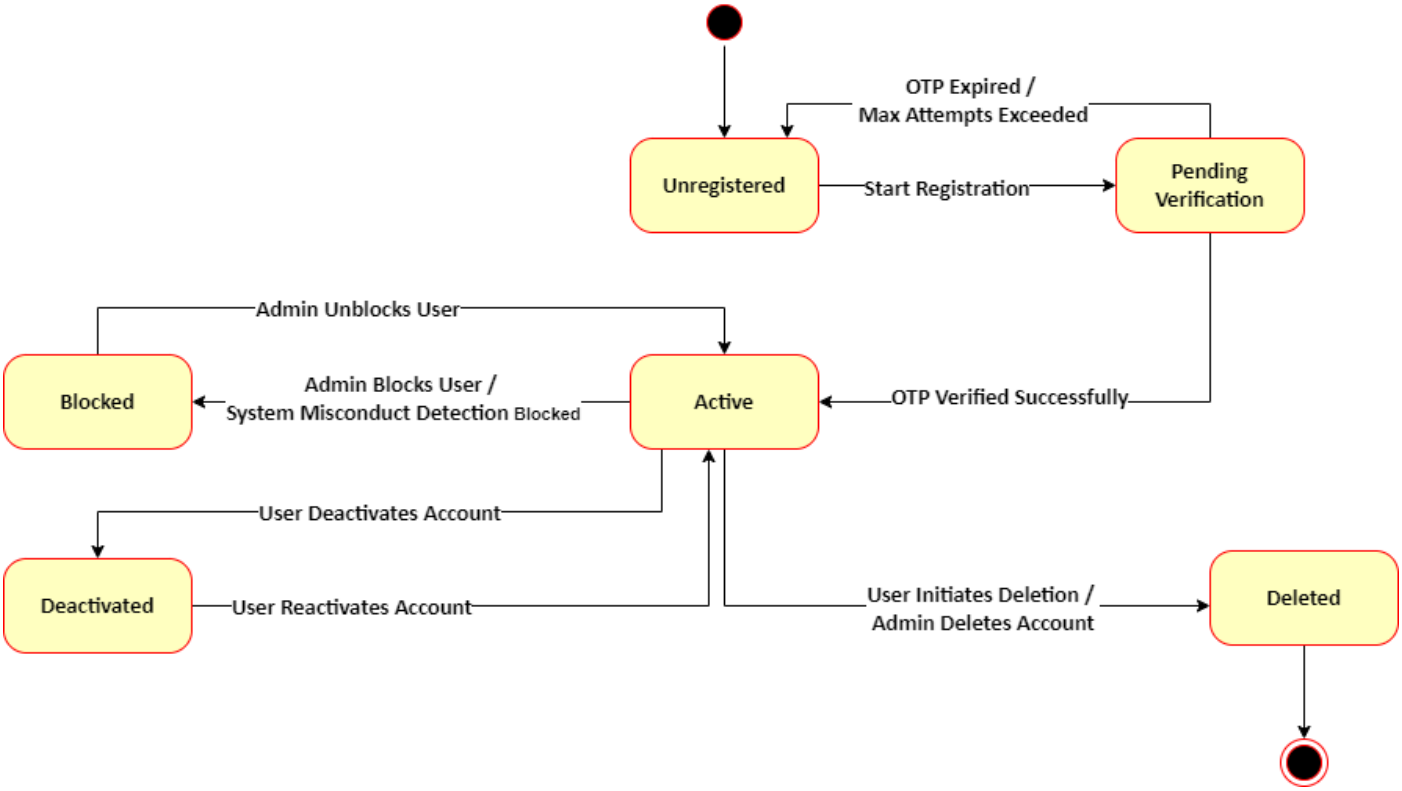
This diagram models the various states a user's account can transition through from creation to deletion, highlighting the events that trigger these changes.

States:

- **Unregistered:** The initial state where no account exists for a user.
- **Pending Verification:** The user has provided a phone number/email, and the system is awaiting verification (e.g., OTP).
- **Active:** The user's account is fully verified and functional, allowing them to use all platform features.
- **Blocked:** The account is temporarily suspended, typically by an administrator, preventing the user from performing most actions.
- **Deactivated:** The user has chosen to temporarily suspend their own account, making it inactive but retaining data for potential reactivation.
- **Deleted:** The account has been permanently removed from the system, usually after a deletion request or retention policy.

Transitions:

1. **Unregistered Start Registration Pending Verification:** A user provides their phone number to begin the registration process.
2. **Pending Verification OTP Verified Successfully Active:** The user successfully enters the correct OTP, verifying their identity.
3. **Pending Verification OTP Expired / Max Attempts Exceeded Unregistered:** The OTP validity period expires, or the user exhausts the maximum number of incorrect OTP attempts.
4. **Active Admin Blocks User / System Misconduct Detection Blocked:** An administrator blocks the user's account due to policy violations or suspicious activity.
5. **Blocked Admin Unblocks User Active:** An administrator reviews and reinstates the user's account.
6. **Active User Deactivates Account Deactivated:** The user chooses to temporarily disable their account.
7. **Deactivated User Reactivates Account Active:** The user logs back in or explicitly requests reactivation of their account.
8. **Active User Initiates Deletion / Admin Deletes Account Deleted:** The user requests permanent deletion, or an administrator manually deletes the account.
9. **Deactivated Data Retention Period Ends / Admin Deletes Account Deleted:** After a period of deactivation, the account is permanently removed based on data retention policies, or an admin deletes it.
10. **Blocked Admin Deletes Account Deleted:** An administrator permanently deletes a blocked account.


OTP Expired /
Max Attempts Exceeded
Unregistered
Start Registration
Pending
Verification
Admin Unblocks User
Blocked
Admin Blocks User /
System Misconduct Detection Blocked
Active
OTP Verified Successfully
User Deactivates Account
Deactivated
User Reactivates Account
User Initiates Deletion /
Admin Deletes Account
Deleted

State Diagram: Message Lifecycle

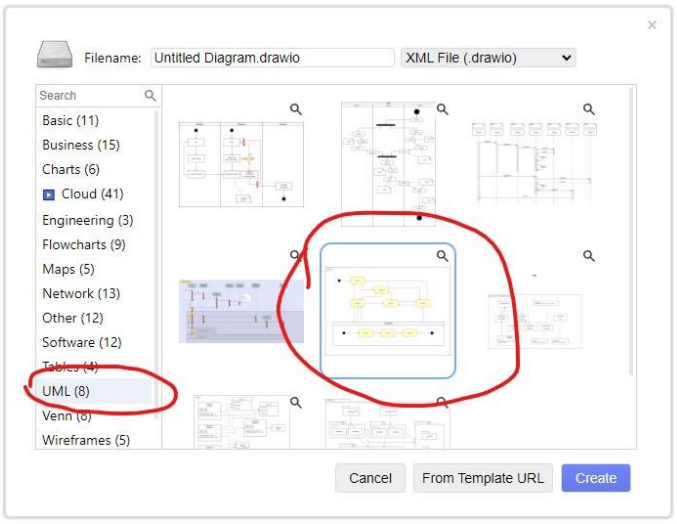
This diagram illustrates the progression of a message through various states from its creation by the sender until it is read or otherwise modified/deleted by the recipient or sender.

States:

- **Draft (Implicit Initial State):** The message is being composed but not yet sent (often managed client-side).
- **Sent:** The message has left the sender's device and been successfully received by the backend server.
- **Delivered:** The message has been successfully pushed from the backend server to the recipient's device(s).
- **Read:** The recipient has opened or viewed the message within the chat interface.
- **Edited:** The message content has been modified by the sender after being sent.
- **Deleted:** The message has been marked for deletion (either for self or for all participants).

Transitions:

1. **Draft User Sends Message Sent:** The user clicks 'send', and the message is transmitted to the backend.
2. **Sent Recipient Client Receives Message Delivered:** The message is successfully received by the recipient's mobile client.
3. **Delivered Recipient Views Message Read:** The recipient opens the chat or scrolls to the message, marking it as read.
4. **Sent Sender Edits Message Edited:** The sender modifies the content of a message that has already been sent (can also transition from Delivered or Read).
5. **Delivered Sender Edits Message Edited:** The sender modifies the content of a message that has been delivered (but not yet read).
6. **Read Sender Edits Message Edited:** The sender modifies the content of a message that has been read.
7. **Sent Sender Deletes Message Deleted:** The sender chooses to delete the message (can be for self or for all).
8. **Delivered Sender Deletes Message Deleted:** The sender deletes a message that has been delivered.
9. **Read Sender Deletes Message Deleted:** The sender deletes a message that has been read.
10. **Edited Sender Deletes Message Deleted:** The sender deletes a message that has been edited.



State Diagram: User Presence / Online Status

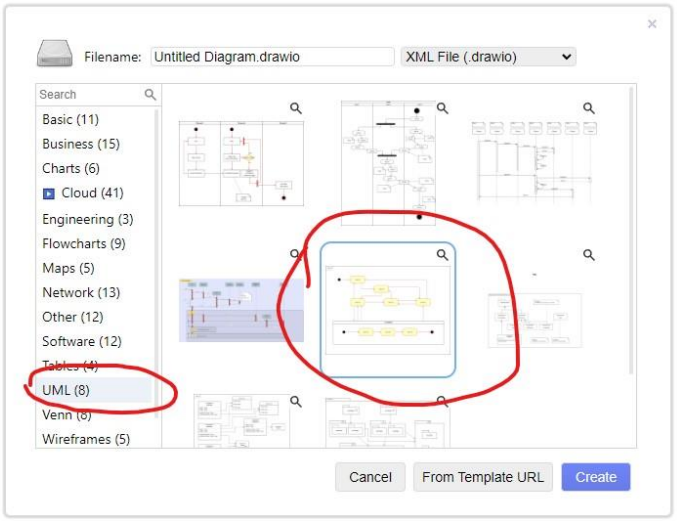
This diagram illustrates the various states a user's presence can be in within the TUASIL system, and the events that trigger transitions between these states, crucial for real-time indicators like "online" or "typing."

States:

- **Offline:** The user is not connected to the TUASIL service, or their client is not active.
- **Online:** The user is actively connected to the TUASIL service and their client is foregrounded or recently active.
- **Away (Idle):** The user is connected but has been inactive for a defined period (e.g., app in background, no recent input).
- **Typing:** The user is actively composing a message in a chat.

Transitions:

1. **Offline User Logs In / App Opens Online:** The user successfully logs into the app, or the app is launched and establishes a connection.
2. **Online Inactivity Timeout Away (Idle):** After a period of no activity (e.g., no screen touch, app in background), the user's status changes to idle.
3. **Away (Idle) User Becomes Active Online:** The user interacts with the app again (e.g., foregrounds it, sends a message).
4. **Online User Starts Typing Typing:** The user begins typing text in a chat input field.
5. **Typing User Stops Typing / Sends Message Online:** The user stops typing, or sends the message, returning to a general online state.
6. **Online User Logs Out / App Closed Offline:** The user explicitly logs out, or the app is fully closed and disconnects.
7. **Away (Idle) User Logs Out / App Closed Offline:** The user logs out or the app closes while they are in an idle state.
8. **Any State Connection Lost Offline:** If the network connection drops unexpectedly, the user's status reverts to offline.



State Diagram: OTP Verification Process

This diagram details the different states involved in the One-Time Password (OTP) verification flow, from its generation to its successful verification, expiration, or failure.

States:

- **Generated:** The OTP has been created by the system.
- **Sent:** The OTP has been dispatched to the user (e.g., via SMS).
- **Awaiting Input:** The system is waiting for the user to enter the received OTP.
- **Verified:** The user has entered the correct OTP within the allowed time and attempts.
- **Expired:** The OTP's validity period has passed before being verified.
- **Failed:** The user entered an incorrect OTP, or exceeded the maximum number of attempts.

Transitions:

1. **Start OTP Requested by User Generated:** A user initiates an action requiring OTP (e.g., registration, password reset), and the system generates an OTP.
2. **Generated OTP Dispatched to User Sent:** The system sends the generated OTP to the user's registered contact method (e.g., phone number).
3. **Sent User Awaits Input Awaiting Input:** The system is now waiting for the user to provide the OTP.
4. **Awaiting Input User Enters Correct OTP Verified:** The user inputs the correct OTP, and the system successfully validates it.

5. **Awaiting Input OTP Validity Timeout Expired:** The time limit for entering the OTP passes before the user provides it.
6. **Awaiting Input User Enters Incorrect OTP Failed:** The user enters an incorrect OTP, leading to a verification failure. (This might loop back to "Awaiting Input" for a few attempts before final "Failed".)
7. **Failed User Requests New OTP Generated:** The user requests a new OTP after a previous attempt failed.
8. **Expired User Requests New OTP Generated:** The user requests a new OTP after the previous one expired.