Implementation

Java :

import java.time.LocalDateTime;

import java.util.List; // For methods that might return lists of objects

// --- Enums for specific types ---

enum ChatType {

PRIVATE,

GROUP,

CHANNEL

}

enum MessageType {

TEXT,

IMAGE,

VIDEO,

FILE,

SYSTEM

}

enum MessageStatus {

SENT,

DELIVERED,

READ

}

enum ChatRole {

MEMBER,

ADMIN,

CREATOR,

SUBSCRIBER

}

enum PrivacySettingValue {

EVERYONE,

MY\_CONTACTS,

NOBODY

}

// --- 1. Class: User ---

public class User {

private int id;

private String phoneNumber;

private String username;

private String firstName;

private String lastName;

private String bio;

private String profilePictureURL;

private boolean isOnline;

private LocalDateTime lastSeenAt;

private UserSettings privacySettings; // Composition/Aggregation: User has UserSettings

private LocalDateTime createdAt;

private LocalDateTime updatedAt;

public User(int id, String phoneNumber, String username, String firstName, String lastName) {

this.id = id;

this.phoneNumber = phoneNumber;

this.username = username;

this.firstName = firstName;

this.lastName = lastName;

this.isOnline = false;

this.createdAt = LocalDateTime.now();

this.updatedAt = LocalDateTime.now();

this.privacySettings = new UserSettings(id); // Initialize default settings

}

// Getters

public int getId() { return id; }

public String getPhoneNumber() { return phoneNumber; }

public String getUsername() { return username; }

public String getFirstName() { return firstName; }

public String getLastName() { return lastName; }

public String getBio() { return bio; }

public String getProfilePictureURL() { return profilePictureURL; }

public boolean isOnline() { return isOnline; }

public LocalDateTime getLastSeenAt() { return lastSeenAt; }

public UserSettings getPrivacySettings() { return privacySettings; }

public LocalDateTime getCreatedAt() { return createdAt; }

public LocalDateTime getUpdatedAt() { return updatedAt; }

// Setters

public void setUsername(String username) { this.username = username; this.updatedAt = LocalDateTime.now(); }

public void setFirstName(String firstName) { this.firstName = firstName; this.updatedAt = LocalDateTime.now(); }

public void setLastName(String lastName) { this.lastName = lastName; this.updatedAt = LocalDateTime.now(); }

public void setBio(String bio) { this.bio = bio; this.updatedAt = LocalDateTime.now(); }

public void setProfilePictureURL(String profilePictureURL) { this.profilePictureURL = profilePictureURL; this.updatedAt = LocalDateTime.now(); }

public void setOnline(boolean online) { isOnline = online; this.updatedAt = LocalDateTime.now(); }

public void setLastSeenAt(LocalDateTime lastSeenAt) { this.lastSeenAt = lastSeenAt; this.updatedAt = LocalDateTime.now(); }

public void setPrivacySettings(UserSettings privacySettings) { this.privacySettings = privacySettings; this.updatedAt = LocalDateTime.now(); }

// Methods

public boolean register(String phoneNumber) {

System.out.println("User registration initiated for phone: " + phoneNumber);

// Logic to send OTP, store temp user data

return true;

}

public boolean authenticate(String otp) {

System.out.println("User authentication with OTP: " + otp);

// Logic to verify OTP and complete authentication

return true;

}

public void updateProfile(String newUsername, String newFirstName, String newLastName, String newBio, String newProfilePicURL) {

System.out.println("Updating user profile for " + this.username);

if (newUsername != null) setUsername(newUsername);

if (newFirstName != null) setFirstName(newFirstName);

if (newLastName != null) setLastName(newLastName);

if (newBio != null) setBio(newBio);

if (newProfilePicURL != null) setProfilePictureURL(newProfilePicURL);

}

public void viewOnlineStatus(User contact) {

System.out.println("Viewing online status for " + contact.getUsername() + " (respecting privacy).");

// Logic to check privacy settings and display status

}

public void viewLastSeen(User contact) {

System.out.println("Viewing last seen for " + contact.getUsername() + " (respecting privacy).");

// Logic to check privacy settings and display last seen

}

public void configurePrivacy(String settingName, PrivacySettingValue value) {

System.out.println("Configuring privacy setting: " + settingName + " to " + value);

// Logic to update specific privacy settings in UserSettings

}

}

// Class: UserSettings (Related to User)

class UserSettings {

private int userId;

private PrivacySettingValue privacyPhoneNumber;

private PrivacySettingValue privacyLastSeen;

private PrivacySettingValue privacyProfilePhoto;

private PrivacySettingValue privacyGroupsAndChannels;

private boolean notificationsPrivateChats;

private boolean notificationsGroupChats;

private boolean notificationsChannels;

public UserSettings(int userId) {

this.userId = userId;

// Default values

this.privacyPhoneNumber = PrivacySettingValue.EVERYONE;

this.privacyLastSeen = PrivacySettingValue.EVERYONE;

this.privacyProfilePhoto = PrivacySettingValue.EVERYONE;

this.privacyGroupsAndChannels = PrivacySettingValue.EVERYONE;

this.notificationsPrivateChats = true;

this.notificationsGroupChats = true;

this.notificationsChannels = true;

}

// Getters

public int getUserId() { return userId; }

public PrivacySettingValue getPrivacyPhoneNumber() { return privacyPhoneNumber; }

public PrivacySettingValue getPrivacyLastSeen() { return privacyLastSeen; }

public PrivacySettingValue getPrivacyProfilePhoto() { return privacyProfilePhoto; }

public PrivacySettingValue getPrivacyGroupsAndChannels() { return privacyGroupsAndChannels; }

public boolean areNotificationsPrivateChatsEnabled() { return notificationsPrivateChats; }

public boolean areNotificationsGroupChatsEnabled() { return notificationsGroupChats; }

public boolean areNotificationsChannelsEnabled() { return notificationsChannels; }

// Setters

public void setPrivacyPhoneNumber(PrivacySettingValue privacyPhoneNumber) { this.privacyPhoneNumber = privacyPhoneNumber; }

public void setPrivacyLastSeen(PrivacySettingValue privacyLastSeen) { this.privacyLastSeen = privacyLastSeen; }

public void setPrivacyProfilePhoto(PrivacySettingValue privacyProfilePhoto) { this.privacyProfilePhoto = privacyProfilePhoto; }

public void setPrivacyGroupsAndChannels(PrivacySettingValue privacyGroupsAndChannels) { this.privacyGroupsAndChannels = privacyGroupsAndChannels; }

public void setNotificationsPrivateChats(boolean notificationsPrivateChats) { this.notificationsPrivateChats = notificationsPrivateChats; }

public void setNotificationsGroupChats(boolean notificationsGroupChats) { this.notificationsGroupChats = notificationsGroupChats; }

public void setNotificationsChannels(boolean notificationsChannels) { this.notificationsChannels = notificationsChannels; }

}

// --- 2. Class: Chat ---

public class Chat {

private int id;

private ChatType chatType;

private String chatName;

private String chatPictureURL;

private String chatDescription;

private String publicLink; // For channels

private int creatorId;

private LocalDateTime createdAt;

public Chat(int id, ChatType chatType, int creatorId) {

this.id = id;

this.chatType = chatType;

this.creatorId = creatorId;

this.createdAt = LocalDateTime.now();

}

// Getters

public int getId() { return id; }

public ChatType getChatType() { return chatType; }

public String getChatName() { return chatName; }

public String getChatPictureURL() { return chatPictureURL; }

public String getChatDescription() { return chatDescription; }

public String getPublicLink() { return publicLink; }

public int getCreatorId() { return creatorId; }

public LocalDateTime getCreatedAt() { return createdAt; }

// Setters

public void setChatName(String chatName) { this.chatName = chatName; }

public void setChatPictureURL(String chatPictureURL) { this.chatPictureURL = chatPictureURL; }

public void setChatDescription(String chatDescription) { this.chatDescription = chatDescription; }

public void setPublicLink(String publicLink) { this.publicLink = publicLink; }

// Methods

public Chat createPrivateChat(int otherUserID) {

System.out.println("Creating private chat with user ID: " + otherUserID);

// Logic to create chat entry in DB

return new Chat(0, ChatType.PRIVATE, this.creatorId); // Placeholder

}

public Chat createGroupChat(String name, List<Integer> initialMemberIDs) {

System.out.println("Creating group chat: " + name + " with " + initialMemberIDs.size() + " members.");

// Logic to create chat and add participants

return new Chat(0, ChatType.GROUP, this.creatorId); // Placeholder

}

public Chat createChannel(String name, String description, boolean isPublic) {

System.out.println("Creating channel: " + name + " (Public: " + isPublic + ")");

// Logic to create channel and set initial properties

return new Chat(0, ChatType.CHANNEL, this.creatorId); // Placeholder

}

public void updateChatDetails(String name, String pictureURL, String description) {

System.out.println("Updating chat details for chat ID: " + this.id);

this.setChatName(name);

this.setChatPictureURL(pictureURL);

this.setChatDescription(description);

// Logic to persist changes to DB

}

public void deleteChat(boolean forUserOnly) {

System.out.println("Deleting chat ID: " + this.id + " (For user only: " + forUserOnly + ")");

// Logic to mark chat as deleted or remove participant entry

}

}

// --- 3. Class: Message ---

public class Message {

private int id;

private int chatId;

private int senderId;

private String content;

private MessageType messageType;

private LocalDateTime sentAt;

private Integer mediaId; // Nullable

private Integer repliedToMessageId; // Nullable

private Integer forwardedFromUserId; // Nullable

private Integer forwardedFromChatId; // Nullable

private LocalDateTime editedAt; // Nullable

private boolean isDeleted;

private int viewCount; // For channels

private MessageStatus status;

public Message(int id, int chatId, int senderId, String content, MessageType messageType) {

this.id = id;

this.chatId = chatId;

this.senderId = senderId;

this.content = content;

this.messageType = messageType;

this.sentAt = LocalDateTime.now();

this.isDeleted = false;

this.viewCount = 0;

this.status = MessageStatus.SENT;

}

// Getters

public int getId() { return id; }

public int getChatId() { return chatId; }

public int getSenderId() { return senderId; }

public String getContent() { return content; }

public MessageType getMessageType() { return messageType; }

public LocalDateTime getSentAt() { return sentAt; }

public Integer getMediaId() { return mediaId; }

public Integer getRepliedToMessageId() { return repliedToMessageId; }

public Integer getForwardedFromUserId() { return forwardedFromUserId; }

public Integer getForwardedFromChatId() { return forwardedFromChatId; }

public LocalDateTime getEditedAt() { return editedAt; }

public boolean isDeleted() { return isDeleted; }

public int getViewCount() { return viewCount; }

public MessageStatus getStatus() { return status; }

// Setters

public void setMediaId(Integer mediaId) { this.mediaId = mediaId; }

public void setRepliedToMessageId(Integer repliedToMessageId) { this.repliedToMessageId = repliedToMessageId; }

public void setForwardedFromUserId(Integer forwardedFromUserId) { this.forwardedFromUserId = forwardedFromUserId; }

public void setForwardedFromChatId(Integer forwardedFromChatId) { this.forwardedFromChatId = forwardedFromChatId; }

public void setEditedAt(LocalDateTime editedAt) { this.editedAt = editedAt; }

public void setDeleted(boolean deleted) { isDeleted = deleted; }

public void setViewCount(int viewCount) { this.viewCount = viewCount; }

public void setStatus(MessageStatus status) { this.status = status; }

// Methods

public boolean send() {

System.out.println("Sending message from user " + senderId + " to chat " + chatId);

// Logic to persist message and notify recipients

return true;

}

public void receive() {

System.out.println("Message ID " + id + " received.");

// Logic to handle message delivery on client side

}

public boolean edit(String newContent) {

System.out.println("Editing message ID " + id);

this.content = newContent;

this.editedAt = LocalDateTime.now();

// Logic to update message in DB and notify chat

return true;

}

public boolean delete(boolean forAll) {

System.out.println("Deleting message ID " + id + " (for all: " + forAll + ")");

this.isDeleted = true; // Soft delete

// Logic to update message in DB and notify chat

return true;

}

public Message reply(int parentMessageID) {

System.out.println("Replying to message ID " + parentMessageID);

// Logic to create a new message with replied\_to\_message\_id

return new Message(0, this.chatId, this.senderId, "Reply to " + parentMessageID, MessageType.TEXT); // Placeholder

}

public Message forward(int targetChatID) {

System.out.println("Forwarding message ID " + id + " to chat " + targetChatID);

// Logic to create a new message with forwarded info

return new Message(0, targetChatID, this.senderId, this.content, this.messageType); // Placeholder

}

public void updateStatus(MessageStatus newStatus) {

System.out.println("Updating status of message ID " + id + " to " + newStatus);

this.status = newStatus;

// Logic to persist status in DB

}

}

// --- 4. Class: Media ---

public class Media {

private int id;

private String filePathOrURL;

private String thumbnailURL;

private Long fileSize; // Use Long for file size in bytes

private String mediaType;

private int uploadedByUserId;

private LocalDateTime uploadedAt;

public Media(int id, String filePathOrURL, String mediaType, int uploadedByUserId) {

this.id = id;

this.filePathOrURL = filePathOrURL;

this.mediaType = mediaType;

this.uploadedByUserId = uploadedByUserId;

this.uploadedAt = LocalDateTime.now();

}

// Getters

public int getId() { return id; }

public String getFilePathOrURL() { return filePathOrURL; }

public String getThumbnailURL() { return thumbnailURL; }

public Long getFileSize() { return fileSize; }

public String getMediaType() { return mediaType; }

public int getUploadedByUserId() { return uploadedByUserId; }

public LocalDateTime getUploadedAt() { return uploadedAt; }

// Setters

public void setThumbnailURL(String thumbnailURL) { this.thumbnailURL = thumbnailURL; }

public void setFileSize(Long fileSize) { this.fileSize = fileSize; }

// Methods

public boolean upload(byte[] fileData) {

System.out.println("Uploading media file of type: " + mediaType);

// Logic to store file data (e.g., to cloud storage) and update filePathOrURL

return true;

}

public byte[] retrieve() {

System.out.println("Retrieving media file from: " + filePathOrURL);

// Logic to retrieve file data

return new byte[0]; // Placeholder

}

public String generateThumbnail() {

System.out.println("Generating thumbnail for media ID: " + id);

// Logic to create thumbnail and update thumbnailURL

this.thumbnailURL = "generated\_thumbnail\_url"; // Placeholder

return thumbnailURL;

}

}

// --- 5. Class: ChatParticipant ---

public class ChatParticipant {

private int id;

private int chatId;

private int userId;

private ChatRole role;

private int unreadCount;

private Integer lastReadMessageId; // Nullable

private LocalDateTime joinedAt;

public ChatParticipant(int id, int chatId, int userId, ChatRole role) {

this.id = id;

this.chatId = chatId;

this.userId = userId;

this.role = role;

this.unreadCount = 0;

this.joinedAt = LocalDateTime.now();

}

// Getters

public int getId() { return id; }

public int getChatId() { return chatId; }

public int getUserId() { return userId; }

public ChatRole getRole() { return role; }

public int getUnreadCount() { return unreadCount; }

public Integer getLastReadMessageId() { return lastReadMessageId; }

public LocalDateTime getJoinedAt() { return joinedAt; }

// Setters

public void setRole(ChatRole role) { this.role = role; }

public void setUnreadCount(int unreadCount) { this.unreadCount = unreadCount; }

public void setLastReadMessageId(Integer lastReadMessageId) { this.lastReadMessageId = lastReadMessageId; }

// Methods

public boolean addMember(int memberUserID) {

System.out.println("Adding user " + memberUserID + " to chat " + chatId);

// Logic to add participant entry

return true;

}

public boolean removeMember(int memberUserID) {

System.out.println("Removing user " + memberUserID + " from chat " + chatId);

// Logic to remove participant entry

return true;

}

public boolean updateRole(int targetUserID, ChatRole newRole) {

System.out.println("Updating role of user " + targetUserID + " in chat " + chatId + " to " + newRole);

// Logic to update role in DB

return true;

}

public boolean leaveChat() {

System.out.println("User " + userId + " leaving chat " + chatId);

// Logic to remove participant entry

return true;

}

public void resetUnreadCount() {

System.out.println("Resetting unread count for user " + userId + " in chat " + chatId);

this.unreadCount = 0;

// Logic to persist change

}

}

// --- 6. Class: Contact ---

public class Contact {

private int id;

private int userId; // The owner of this contact entry

private int contactUserId; // The user added as a contact

private String aliasName; // Optional

private LocalDateTime createdAt;

public Contact(int id, int userId, int contactUserId) {

this.id = id;

this.userId = userId;

this.contactUserId = contactUserId;

this.createdAt = LocalDateTime.now();

}

// Getters

public int getId() { return id; }

public int getUserId() { return userId; }

public int getContactUserId() { return contactUserId; }

public String getAliasName() { return aliasName; }

public LocalDateTime getCreatedAt() { return createdAt; }

// Setters

public void setAliasName(String aliasName) { this.aliasName = aliasName; }

// Methods

public boolean addContact(int targetUserID) {

System.out.println("User " + userId + " adding contact " + targetUserID);

// Logic to create contact entry in DB

return true;

}

public List<Contact> viewContacts() {

System.out.println("Viewing contacts for user " + userId);

// Logic to fetch contacts from DB

return List.of(); // Placeholder

}

public void assignAlias(String alias) {

System.out.println("Assigning alias '" + alias + "' to contact " + contactUserId);

this.aliasName = alias;

// Logic to persist alias

}

public boolean blockContact() {

System.out.println("User " + userId + " blocking contact " + contactUserId);

// Logic to create blocked\_users entry

return true;

}

}

// --- 7. Class: Notification ---

public class Notification {

private int id;

private int recipientUserID;

private String message;

private LocalDateTime timestamp;

private String eventType;

private Integer relatedChatID; // Nullable

private boolean isRead;

public Notification(int id, int recipientUserID, String message, String eventType) {

this.id = id;

this.recipientUserID = recipientUserID;

this.message = message;

this.eventType = eventType;

this.timestamp = LocalDateTime.now();

this.isRead = false;

}

// Getters

public int getId() { return id; }

public int getRecipientUserID() { return recipientUserID; }

public String getMessage() { return message; }

public LocalDateTime getTimestamp() { return timestamp; }

public String getEventType() { return eventType; }

public Integer getRelatedChatID() { return relatedChatID; }

public boolean isRead() { return isRead; }

// Setters

public void setRelatedChatID(Integer relatedChatID) { this.relatedChatID = relatedChatID; }

public void setRead(boolean read) { isRead = read; }

// Method

public boolean send() {

System.out.println("Sending notification to user " + recipientUserID + ": " + message);

// Logic to push notification via external services (FCM/APNS)

return true;

}

}

**Database code**

CREATE DATABASE IF NOT EXISTS tuasil\_messaging;

USE tuasil\_messaging;

-- 1. Class: User

-- Represents individual registered users of the platform

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

phone\_number VARCHAR(20) NOT NULL UNIQUE, -- Used for registration and login

username VARCHAR(50) UNIQUE, -- Optional, but must be unique if set (FR-UM-4)

first\_name VARCHAR(100),

last\_name VARCHAR(100),

bio TEXT,

profile\_picture\_url VARCHAR(255),

is\_online BOOLEAN DEFAULT FALSE, -- FR-UM-7

last\_seen\_at DATETIME, -- FR-UM-8

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- 1.a. Related to User: Class UserSettings (Implicit from FR-SEC-x and references)

-- Stores user-specific privacy and notification preferences

CREATE TABLE user\_settings (

user\_id INT PRIMARY KEY,

privacy\_phone\_number ENUM('everyone', 'my\_contacts', 'nobody') DEFAULT 'everyone', -- FR-SEC-3

privacy\_last\_seen ENUM('everyone', 'my\_contacts', 'nobody') DEFAULT 'everyone', -- FR-SEC-4

privacy\_profile\_photo ENUM('everyone', 'my\_contacts', 'nobody') DEFAULT 'everyone', -- FR-SEC-5

privacy\_groups\_and\_channels ENUM('everyone', 'my\_contacts') DEFAULT 'everyone', -- FR-SEC-6

notifications\_private\_chats BOOLEAN DEFAULT TRUE, -- FR-NOTIF-1

notifications\_group\_chats BOOLEAN DEFAULT TRUE, -- FR-NOTIF-2 (example settings)

notifications\_channels BOOLEAN DEFAULT TRUE, -- FR-NOTIF-2 (example settings)

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- 1.b. Related to User: Sessions (Implicit from NFR-NOTIF-1 and sessions table reference)

-- Stores active user sessions for push notifications

CREATE TABLE sessions (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

device\_token VARCHAR(255) NOT NULL, -- Device-specific token for push notifications (FCM, APNS)

is\_active BOOLEAN DEFAULT TRUE,

last\_active\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

UNIQUE (user\_id, device\_token) -- A user can only have one session per device token

);

-- 2. Class: Chat

-- Represents conversation threads: private, group, or channel

CREATE TABLE chats (

id INT AUTO\_INCREMENT PRIMARY KEY,

chat\_type ENUM('private', 'group', 'channel') NOT NULL,

chat\_name VARCHAR(255), -- FR-CH-3, FR-CH-8 (for groups/channels)

chat\_picture\_url VARCHAR(255), -- FR-CH-3, FR-CH-8

chat\_description TEXT, -- FR-CH-3, FR-CH-8

public\_link VARCHAR(255) UNIQUE, -- FR-CH-8 (for public channels)

creator\_id INT NOT NULL, -- Who created the chat/group/channel

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (creator\_id) REFERENCES users(id) ON DELETE RESTRICT -- Creator cannot be deleted if chat exists

);

-- 5. Class: ChatParticipant

-- Represents a user's involvement in a specific chat

CREATE TABLE chat\_participants (

id INT AUTO\_INCREMENT PRIMARY KEY,

chat\_id INT NOT NULL,

user\_id INT NOT NULL,

role ENUM('member', 'admin', 'creator', 'subscriber') DEFAULT 'member', -- FR-CH-3, FR-CH-5, FR-CH-8

unread\_count INT DEFAULT 0, -- FR-MSG-13

last\_read\_message\_id INT, -- FR-MSG-13 (Can be NULL)

joined\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (chat\_id, user\_id), -- A user can only be a participant once per chat

FOREIGN KEY (chat\_id) REFERENCES chats(id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- 4. Class: Media

-- Stores information about media files (images, videos, files)

CREATE TABLE media (

id INT AUTO\_INCREMENT PRIMARY KEY,

file\_path\_or\_url VARCHAR(255) NOT NULL, -- FR-MEDIA-2, FR-MEDIA-3

thumbnail\_url VARCHAR(255), -- FR-MEDIA-4

file\_size BIGINT, -- NFR-SCAL-3 implicitly

media\_type VARCHAR(50), -- E.g., 'image/jpeg', 'video/mp4'

uploaded\_by\_user\_id INT NOT NULL,

uploaded\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (uploaded\_by\_user\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- 3. Class: Message

-- Represents a unit of communication within a chat

CREATE TABLE messages (

id INT AUTO\_INCREMENT PRIMARY KEY,

chat\_id INT NOT NULL,

sender\_id INT NOT NULL,

content TEXT, -- FR-MSG-1, FR-MSG-3 (for caption)

message\_type ENUM('text', 'image', 'video', 'file', 'system') NOT NULL,

sent\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- NFR-PERF-1, FR-MSG-12

media\_id INT, -- FR-MSG-3 (Nullable for text messages)

replied\_to\_message\_id INT, -- FR-MSG-7 (Nullable)

forwarded\_from\_user\_id INT, -- FR-MSG-8 (Nullable)

forwarded\_from\_chat\_id INT, -- FR-MSG-8 (Nullable)

edited\_at DATETIME, -- FR-MSG-9 (Nullable)

is\_deleted BOOLEAN DEFAULT FALSE, -- FR-MSG-10 (Soft delete)

view\_count INT DEFAULT 0, -- FR-MSG-14 (For channels)

FOREIGN KEY (chat\_id) REFERENCES chats(id) ON DELETE CASCADE,

FOREIGN KEY (sender\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (media\_id) REFERENCES media(id) ON DELETE SET NULL, -- Media can exist without a message or be deleted separately

FOREIGN KEY (replied\_to\_message\_id) REFERENCES messages(id) ON DELETE SET NULL, -- Self-referencing

FOREIGN KEY (forwarded\_from\_user\_id) REFERENCES users(id) ON DELETE SET NULL, -- If forwarded, link to original sender

FOREIGN KEY (forwarded\_from\_chat\_id) REFERENCES chats(id) ON DELETE SET NULL , -- If forwarded, link to original chat

-- Expert Engineer's Perspective: Index for efficient message retrieval by chat and time

INDEX idx\_messages\_chat\_id\_sent\_at (chat\_id, sent\_at DESC)

);

-- 6. Class: Contact

-- Represents a user's contact list

CREATE TABLE contacts (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL, -- The user who owns this contact entry

contact\_user\_id INT NOT NULL, -- The user who is added as a contact

alias\_name VARCHAR(100), -- FR-CM-2 (Optional)

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (user\_id, contact\_user\_id), -- A user can only add another user as a contact once

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (contact\_user\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- Related to Contact: BlockedUser (Implicit from FR-CM-4, FR-CM-5 and blocked\_users table reference)

-- Stores pairs of users where one has blocked the other

CREATE TABLE blocked\_users (

blocker\_id INT NOT NULL, -- The user who initiated the block

blocked\_id INT NOT NULL, -- The user who is blocked

blocked\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (blocker\_id, blocked\_id), -- A user can only block another user once

FOREIGN KEY (blocker\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (blocked\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- 7. Class: Notification (This refers to general in-app notifications, not necessarily push tokens)

-- Stores records of notifications sent within the system

CREATE TABLE notifications (

id INT AUTO\_INCREMENT PRIMARY KEY,

recipient\_user\_id INT NOT NULL, -- FR-NOTIF-1

message TEXT NOT NULL, -- FR-NOTIF-1

event\_type VARCHAR(100), -- E.g., 'new\_message', 'group\_add', 'admin\_change'

related\_chat\_id INT, -- Optional: Link to the chat if notification is chat-related

is\_read BOOLEAN DEFAULT FALSE,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (recipient\_user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (related\_chat\_id) REFERENCES chats(id) ON DELETE SET NULL

);

-- Add indexes for common lookup fields to improve performance

CREATE INDEX idx\_users\_phone\_number ON users(phone\_number);

CREATE INDEX idx\_users\_username ON users(username);

CREATE INDEX idx\_chats\_chat\_type ON chats(chat\_type);

CREATE INDEX idx\_chat\_participants\_user\_id ON chat\_participants(user\_id);

CREATE INDEX idx\_messages\_sender\_id ON messages(sender\_id);

CREATE INDEX idx\_messages\_media\_id ON messages(media\_id);

CREATE INDEX idx\_contacts\_user\_id ON contacts(user\_id);

CREATE INDEX idx\_notifications\_recipient\_user\_id ON notifications(recipient\_user\_id);