Team Name: DAAP-SecDov-518

People: Aryan Kumar, Darshil Jayani, Alejandro Miller-Gonzalez, Partho Bhattacharya

Overview:

This project focuses on building a secure, web-based messenger client with end-to-end encryption. It provides authenticated users with a platform to send and receive real-time messages. Core features include multi-factor authentication (MFA) for login and logout, encrypted message exchange with notifications, message history and backup, message and account deletion, group creation and management, audit logging, and profile updates.

Use Case Diagram

Use Case Tables

Use Case Name	**Create Account**
Actors	User
Preconditions	
Goal	Let user create an account with which to access the messaging service. A user can have multiple accounts
Scenario	1. User selects option to create an account 2. User selects account name and password, along with a valid email and phone number (if multifactor authentication is desired by the user) 3. User goes through reCAPTCHA to attempt to protect account creation process against malicious bots 4. User's account gets updated to the newly created account
Exceptions	● User fails reCAPTCHA process ● User does not have a valid email

Use Case Name	**Make Account**
Actors	Server
Preconditions	● Account name is unique ● Account name doesn't include disallowed characters
Goal	User's desired account is registered to the database
Scenario	1. Server receives instruction to create account from user 2. A new account under the given name is appended to the database 3. User's active session is logged into the newly created account
Exceptions	● Account name isn't unique/already exists ● Account name includes disallowed characters ● Server fails to update database

Use Case Name	**Login**
Actors	User, Authentication Service

Preconditions	● User has an account with the authentication service
Goal	Authenticate users before they can access private, account specific information and create/send messages as the account holder
Scenario	1. User navigates to the web home page of the messaging client and selects option to log in 2. User enters in their username, password, and goes through reCAPTCHA before continuing 3. If multifactor authentication is turned on for this account, a message is sent to the user's phone with a code that they will need to enter on the page to finish logging in 4. If the user information is valid, the user will be logged into the messaging service under their user account 5. If the user information is invalid, the user is not logged in to the messenger service. There is a limit to the total number of login attempts within a period of time.
Exceptions	● Authentication service fails for any reason

Use Case Name	**Retrieve Account**
Actors	Server
Preconditions	● Given account exists within the server's database
Goal	Update the authenticated user's current session to provide access into the account
Scenario	1. Server receives the authenticated user's credentials 2. Server's database is searched for the unique account that corresponds to the user's given credentials 3. User's session is provided access and control over the validated account
Exceptions	● Server fails to retrieve the user's account from the database

Use Case Name	**Password Reset**
Actors	User, Server
Preconditions	● User provides a valid email address, and an account tied to that email exists
Goal	User is able to initiate a password reset for a desired account with its associated email

Scenario	1. User selects “forgot password” option on login page 2. User provides a valid email address for an associated account 3. Server generates a temporary token and sends it to the given email along with a link to a password reset page 4. User enters a new password and the token on the linked reset page 5. Account’s associated password is changed if the token is valid
Exceptions	● No account associated with provided email ● Server fails to respond to user’s request ● User-entered temporary token is invalid (expired/incorrect)

Use Case Name	**Remove Password**
Actors	Server
Preconditions	● Server has validated user’s password reset request with temporary token
Goal	Update the account’s password to new one chosen by user
Scenario	1. Server receives validated request to change password, along with the desired new password 2. Server’s database overwrites user’s old password with new one
Exceptions	● User’s new desired password contains disallowed characters ● The server’s database fails to update password

Use Case Name	**Profile Management**
Actors	User
Preconditions	● Already logged in to the account.
Goal	The user can change their email ID and phone number
Scenario	1. User selects the Profile Management section 2. User selects the email ID or phone number field to update information with an editable button. 3. User enters a new email ID/phone number.

Exceptions	● Already have an account with the entered email ID / phone number. ● User enters an invalid character

Use Case Name	**Update Info**
Actors	Server
Preconditions	● Given account exists within the server's database
Goal	Update database with account associated information that has changed, like email ID and phone number
Scenario	1. Server receives a validated request to update account associated information (either phone number, email, or both) 2. Server's database overwrites user's old account associated information (either phone number, email, or both) with updated information
Exceptions	● User's new entered email is invalid ● User's new entered phone number is invalid ● Server's database fails to update account associated information

Use Case Name	**Send Message**
Actors	User
Preconditions	● User is logged into an account
Goal	User may send a message to another account
Scenario	1. User selects "new message" option 2. User defines the destination account or group 3. User writes their desired message 4. User sends the message with the "send" option
Exceptions	● Destination account is invalid (doesn't exist or the destination is the user) ● Message contains disallowed characters

Use Case Name	**Read Message**
Actors	User
Preconditions	● User is logged into an account

Goal	To view any messages sent in conversation with any other users or groups where messages have been successfully sent
Scenario	1. By the list of message chats in the application that the user can view, the user selects the “Read Message” option 2. User views the list of messages sent between all parties in the conversation or group (if in a group, user only sees messages sent after the user joined the group based on the time stamp signatures of the message and of the joining time.
Exceptions	● If no messages have been sent in a group since an invite was accepted, no messages will show (even if messages were previously sent)

Use Case Name	**Store Message**
Actors	Server
Preconditions	● Message was sent to and from a valid user
Goal	Update server’s database with a newly sent user’s message
Scenario	1. Server receives a user’s message 2. Server updates database with the new message, the sender of the message, and the time at which the message was sent
Exceptions	● Server fails to update database with new message

Use Case Name	**Create Group**
Actors	User, Server
Preconditions	● User is logged into account
Goal	Allow users to create secure group chats that have unique encryption keys for all participants.
Scenario	1. User selects “new group” option 2. User enters name for group with valid characters 3. User enters other account usernames for other users they would like to add to the group 4. Server creates a unique group tag id (separate from the group name) that distinguishes it from other groups and is used in other group operations to find the group. 5. Server sends notification to all users on list with valid usernames that they have been invited to a group

	6. The group name is added as a valid destination option for the “new message” option for this user
Exceptions	● Group name has invalid characters

Use Case Name	**Store Group**
Actors	Server
Preconditions	● Create Group request was sent to the server by a valid user account
Goal	Create a new entry in the groups database that includes all the group specific information inputted by the user
Scenario	1. Server receives a validated request to create a group 2. Server creates a new entry in the groups database that includes- the names of valid user accounts that have been invited to the group, the names of valid user accounts who have accepted being invited to the group including the owner (and all their unique encryption keys), the time at which each user joined (or accepted the invite), the name of the group, and the admin of the group (set to be the creator of the group)
Exceptions	● Server fails to update database with new group information ● Group information contains invalid characters (invalid users are fine as invites to group simply won’t be sent to usernames that are invalid)

Use Case Name	**Accept Group**
Actors	User
Preconditions	● User is logged into account
Goal	Accept an invitation to join a group created by another user (or to join one that the user created but previously left)
Scenario	1. User selects option to “Accept Invite” in their notifications channel which communicates the name of the group and which user in the group invited the user
Exceptions	User didn’t find the invitation.

Use Case Name	**Leave Group**

Actors	User
Preconditions	● User is logged into account
Goal	Leave a group that you are currently a part of.
Scenario	1. By the list of message chats in the application that the user can view, the user selects the “Leave Group” option next to a group message chat
Exceptions	● If the user is the only admin of the group, the user is given a message to first update the group to have new admins or to delete the group instead.

Use Case Name	**Update Group**
Actors	User, Server
Preconditions	● User is logged into account and is the admin of the group to modify
Goal	Modify a group that you are currently a part of (and are the admin of)
Scenario	1. By the list of message chats in the application that the user can view, the user selects the “Modify Group” option next to a group message chat 2. The user is navigated to a screen that shows active users, invited users, the group name, and the admins of the group. The admin can remove active users, invite users, uninvite users, modify the group name, or update who the admins of the group are (but cannot remove themselves as admins if there are no other admins for the group and cannot remove other admins).
Exceptions	● If the user is the only admin of the group, the user is given a message to first update the group to have new admins or to delete the group instead.

Use Case Name	**Modify Group**
Actors	Server
Preconditions	● Modify group request was sent by a validated user account (can include leave group, accept group, or update group requests)
Goal	To modify the entry in the groups database that includes all the

	group specific information inputted by the user
Scenario	1. Server receives a validated request to modify a group 2. Server modifies an existing entry in the groups database that includes- adding or removing a username that was invited to a group but has not yet accepted, adding or removing a username of a valid user account which has accepted being invited to the group excluding the user themselves, changing the name of the group, and changing the admin of the group. If users are removed from the active user list, their associated time information of when they joined are also removed. If a user joins the active user list after accepting an invite, their associated time information of when they joined is also added.
Exceptions	● Invalid user accounts are provided ● Server fails to update database with updated group information

Use Case Name	**Delete Message**
Actors	User
Preconditions	● A message sent or received by the user exists
Goal	User may delete a desired message
Scenario	1. User selects a message 2. User selects the “delete message” option 3. Message is removed from user’s view
Exceptions	

Use Case Name	**Remove Message**
Actors	Server
Preconditions	● A message sent or received by the user exists
Goal	Server removes a specified message from its database
Scenario	1. Server receives instruction from user to delete a message 2. Server locates the message within its database 3. Server removes the message entry from its database
Exceptions	● Server fails to delete the message from the database

Use Case Name	**Delete Group**
Actors	User
Preconditions	● User is an admin of the group to be deleted
Goal	To let the user delete a group that they are a part of
Scenario	1. By the list of message chats in the application that the user can view, the user selects the “Delete Group” option next to a group message chat 2. User receives a prompt asking them to confirm if they truly wish to delete the group 3. Once the group has been deleted, a notification is sent to all active users up to that point that the group has been deleted
Exceptions	● User is not validated

Use Case Name	**Remove Group**
Actors	Server
Preconditions	● Delete Group request was sent to the server by a valid user account that is the admin of the group to be deleted
Goal	Delete an entry in the groups database that includes all the group specific information inputted by users
Scenario	1. Server receives a validated request to delete a group 2. Server removes the entry for the group in its database, including all group associated information.
Exceptions	● Server fails to update database with updated group information

Use Case Name	**File Upload**
Actors	User, Server
Preconditions	● User is logged in to the account
Goal	User can upload a file (image, document) to the server.
Scenario	1. User navigates to group and select the upload file option. 2. User searches and selects a file from local storage. 3. System checks for the correct size and type of file 4. User clicks on the send button, and the system sends

	the file to server 5. Server confirms successful upload with ‘upload successful’ notification
Exceptions	● File format not supported and size limit exceed. ● Server fails to upload the file.

Use Case Name	**Store File**
Actors	Server
Preconditions	● User request received for file upload
Goal	Store the uploaded file securely on the server
Scenario	1. Server received a request from the user. 2. Server performs checks on file size and type. 3. Server saves the file. 4. Server sends a confirmation notification message to user.
Exceptions	● User uploaded an unsupported file type or exceeded the limt of file size. ● Server fails to save the file. ● Storage is full or inaccessible.

Use Case Name	**Backup Message**
Actors	User
Preconditions	● User logged in to the account ● User has a history of chat in their account.
Goal	Users are able to back up their messages
Scenario	1. User selects a backup messages option from profile management. 2. User confirms the backup request. 3. System retrieves all the messages from server. 4. System creates a backup file and sendt to the server.
Exceptions	● User has no messages for backup. ● Backup location is full of space. ● Backup stops due to system errors.

Use Case Name	**Save Message**
Actors	Server

Preconditions	● User request for backup messages. ● Backup file has been created in system.
Goal	Save backup file to server.
Scenario	1. System sends the backup file to the server 2. Server stores a backup file to secure storage. 3. Server update the database with backup detail linked with the user account (username, timestamp, file location). 4. Server sends a notification to the user after successfully saving the backup file.
Exceptions	● Storage limit exceeded. ● Server fails to store backup file. ● System down while sending backup file.

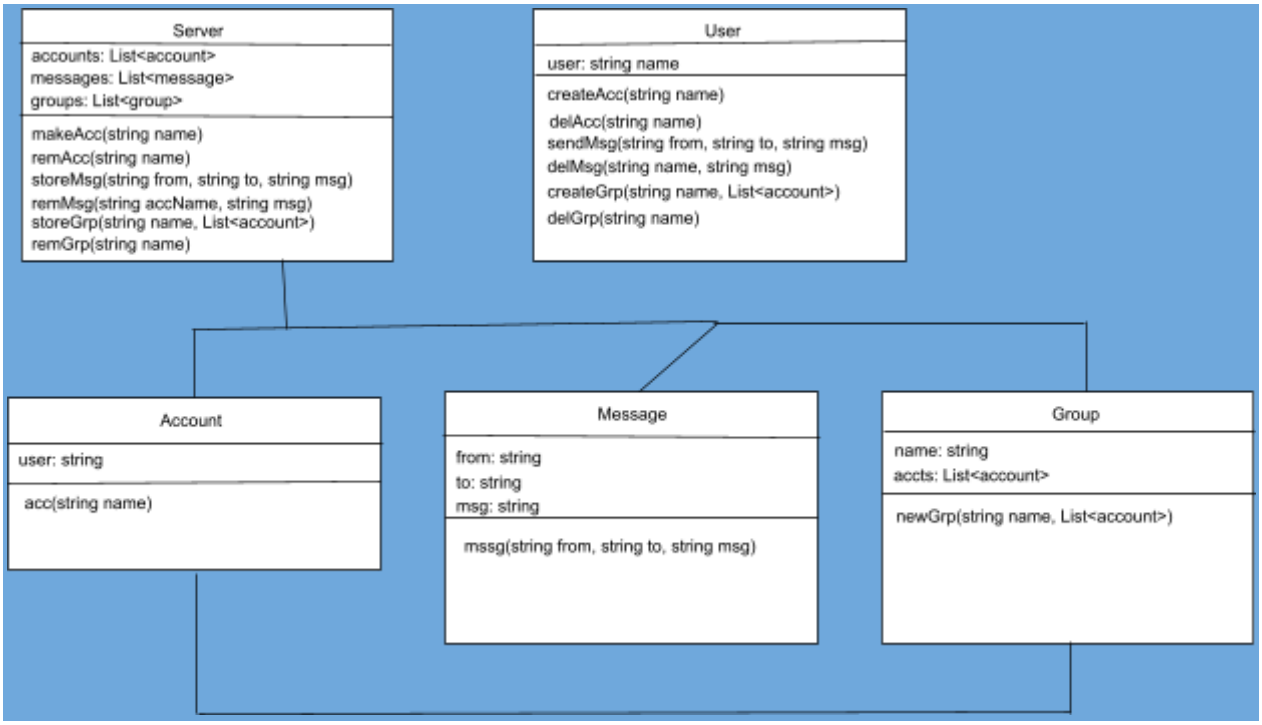
Use Case Name	**Account Activity**
Actors	User
Preconditions	● User is logged into an account
Goal	User can view a log of their account activity that cannot be modified by them
Scenario	1. User selects “Account Activity” option 2. User may read a complete audit log of the account’s activity history
Exceptions	

Use Case Name	**Audit Times**
Actors	Server
Preconditions	● An auditable action was performed by a user
Goal	To maintain a complete audit log of all activity associated with an account that is visible to an authenticated user of the account
Scenario	1. User performs an auditable event (login, logout, message sent, file upload, etc.) 2. “Account activity” is updated within the server’s database with a new entry of the timestamped event
Exceptions	● Server fails to update the audit log within the database

Use Case Name	**Logout**
Actors	User
Preconditions	● User is logged into an active session
Goal	Sign the user out of their current session
Scenario	1. User selects “sign out” option 2. User’s active session is updated to remove access and control over account
Exceptions	

Use Case Name	**Clear Session**
Actors	Server
Preconditions	● User is logged in. ● User request for log out session.
Goal	Terminate the user’s active session and log out securely.
Scenario	1. Server successfully received log out request. 2. Server terminate current session for requested user. 3. Server confirms session log out to user via notification. 4. System redirect user to login screen.
Exceptions	● The timestamp already expired before the logout request. ● Server fails to clear the session.

UML Class Diagrams



Quality Plan

Security Goals

Confidentiality

- Users should only be able to view the messages sent by them to others or messages sent to them by others
    - Message history will show older messages sent to and from the user.
- Users should be able to upload and share small files with whom they want to, and the other users shouldn't be able to tell or see what the contents of the file are.
    - The file upload and sharing will be done safely with encryption so the contents are not seen by a third party.
- Users should only be able to see groups they are invited to and accepted into.
    - Groups will have a list of members that are in them that can't be seen by users outside of the group.
- Only group owners should be able to delete the groups, message owners should be able to delete their messages, and account owners should be able to delete their accounts.
    - Only authorized users can see and delete groups, messages and accounts that will be the owners of all.

Integrity

- File upload will be checksumed and checked on arrival
    - Checksum will be calculated and sent with the file and will be checked on arrival to make sure that the file wasn't tampered with.
- Users won't be able to edit or delete messages sent by others.
    - Messages should not be editable or deletable by people who didn't send them.
- Users outside of the group should not be able to change the users in a group
    - Only users inside the group can leave or join a group.
    - Only the group owner can invite new people to the group.
    - Users on the outside of the group can't join or tamper with the users inside the group.

Availability

- Users should be able to send messages at all times to whoever they want
    - Messaging is available to users whenever they require it.
- Users should be able to upload and send files at all times to whoever they want.
    - File uploading and sharing are available to users whenever they require.
- Users outside of a group can't change group settings and members
    - Users outside of the group can't see the member inside, and the group is not available to them unless the group owner invites them

Security Metrics

Confidentiality -

- Percentage of messages that are viewed by a user other than the intended recipient or recipients. Note that this also includes messages in groups that are either viewable by users not in that group OR messages in groups that are viewable by users in that group, but who became active users (accepted invites) of that group after those messages had been sent (as users should not have access to messages from before they joined the group). Users should also not have access to group messages once they leave a group.
    - Target will be 0% and will be greater than 0% if any vulnerabilities are discovered

Integrity -

- Percentage of messages that have their content modified in any way (either by accident, through malicious means, or through failures/bugs) after being sent by a sender
    - Target will be 0% and will be greater than 0% if any vulnerabilities are discovered
- Percentage of messages that have their sender field modified in any way (either by accident, through malicious means, or through failures/bugs) after being sent by a sender to appear as if they are coming from another user

    - Target will be 0%, and will be greater than 0% if any vulnerabilities are discovered

Availability

- Server downtime percentage
    - Target will be 0% and will be greater than 0% if server cannot defend against denial of service attacks, or if it cannot be powered on
- Percentage of server database updates that fail to complete
    - Target will be 0% and will be greater than 0% if the database does not properly update after applicable user actions (sending messages, uploading files, deleting messages, audit logging, etc.). for any reason
- Percentage of database deletions done by attackers
    - Target will be 0% and will be greater than 0% if an unauthorized user can exploit a vulnerability to delete elements from the database