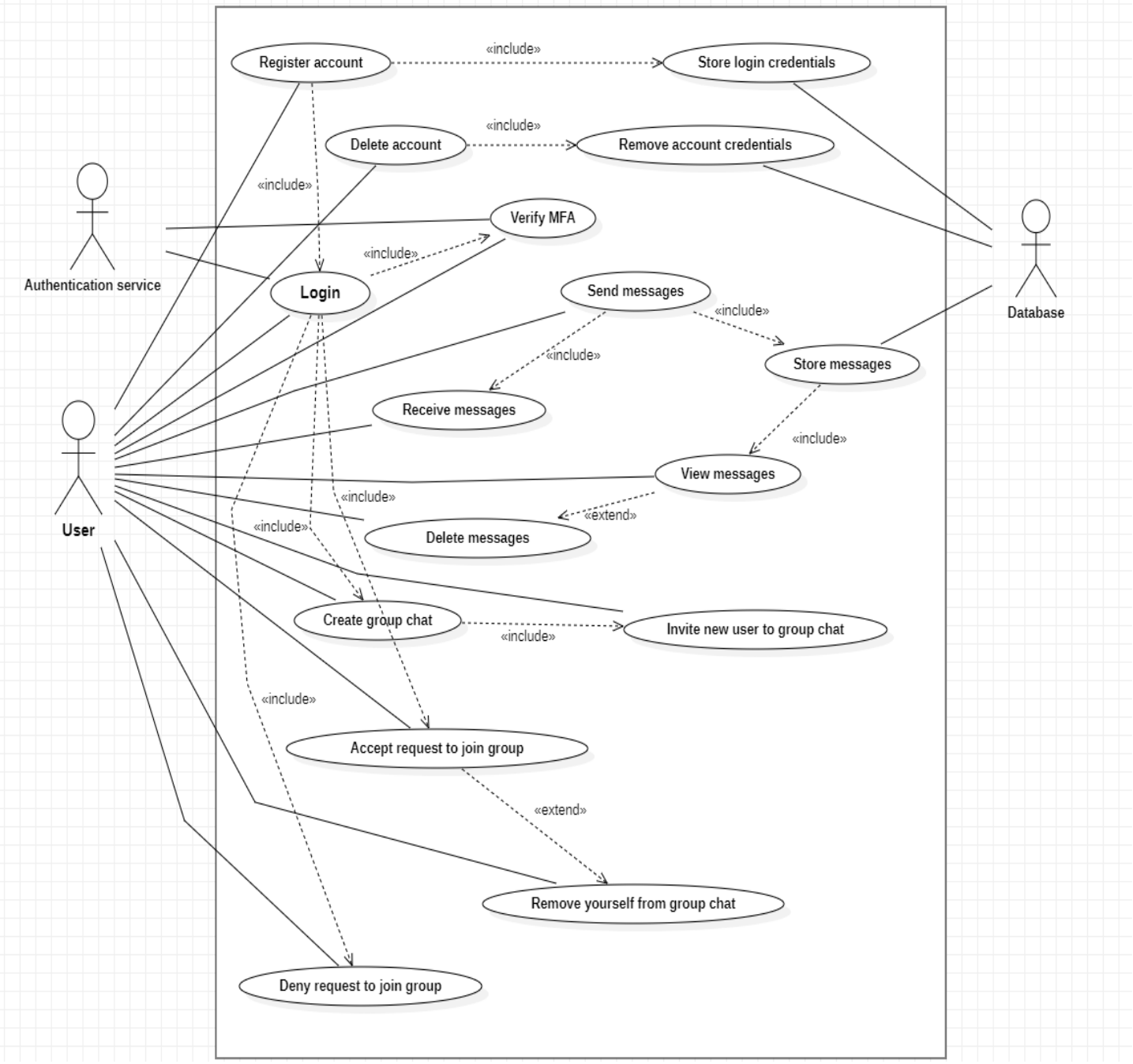
CptS 428/528 Software Security and Reverse Engineering
Fall 2023
Team: Undergraduate ABFMJ-SecureDove
Project Deliverable 2 Assure

Team Members: Muhammad Ali Dzulfiqar, Flavio Alvarez Penate, Phearak Both Bunna, Jaysen Anderson, Mitchell Kolb

Use Case Diagram:



Use Case Tables:

Use Case Name	**Creating Account**
Actors	User, Account Service
Preconditions	● The site is able to successfully load and user has clicked on the register account button on home page
Goal	Creates a new account for the user so they are able to use the messenger site. One device/browser instance per user
Scenario	1. User clicks the “register for new account” button 2. User enters account username 3. User enters account password 4. Website confirms creation of account 5. Page is reloaded with user signed in
Exceptions	● Web page has no account logged in ● During creation account name is unique ● During creation password is verified to contain particular characters and symbols

Use Case Name	**Logging into Account**
Actors	User, Authentication Service
Preconditions	● User has created an account and has clicked on the login button
Goal	Get the user logged into the correct account
Scenario	1. User arrives on the homepage and clicks the login button 2. User enters their credentials 3. Authentication service performs validation 4. Authentication service informs user on if they were able to login or not 5. If user logs in successfully, the user can access their messenger account 6. If user fails to login, they cannot access any messenger account
Exceptions	● Authentication service fails

Use Case Name	**Deleting Account**
Actors	User, Database

Preconditions	● User has an account and is logged in
Goal	User deletes their messenger account
Scenario	1. User logs into their messenger account 2. User clicks the “Delete account” button 3. User must confirm the deletion of the account by typing in the username associated with the account 4. User's account is removed from the database
Exceptions	● User fails to type the username in correctly

Use Case Name	**Verifying MFA for Account**
Actors	User, Authentication service
Preconditions	● User has logged in and clicked the multi factor authentication button
Goal	User sets up multi factor authentication for logging into their account
Scenario	1. User logs into their messenger account 2. User clicks the “Setup MFA” button 3. User enters their phone number 4. Authentication service sets up the multi factor authentication
Exceptions	● Multi factor authentication has already been setup for the account ● The phone number used has already been used on a different account

Use Case Name	**Sending Messages**
Actors	User
Preconditions	● User is logged into their account
Goal	User can send a message to a chat
Scenario	1. User logs in 2. User either opens a chat or clicks the “Send new message” button 3. If the user clicks the “Send new message” button, the User must select a recipient of the message 4. User types the message 5. User clicks the “Send” button

Exceptions	● Recipient does not exist ● Message is too long

Use Case Name	**Storing Send Messages**
Actors	Database
Preconditions	● A user has sent a new message and the database has received it
Goal	Update the database with the newly sent message
Scenario	1. Database receives a new message 2. Database records the new message in the database
Exceptions	● Database error

Use Case Name	**Receiving Messages**
Actors	User
Preconditions	● The database has confirmed and linked a sent/stored message to the intended message recipient.
Goal	To update the user's chat log with new messages
Scenario	1. The user logs in 2. The message exists and is sent 3. The user is idle, after a preset amount of time the interface pings the database for new messages. 4. If there is new messages the interface is reloaded to show them
Exceptions	● When the interface pings the database and there is no new messages

Use Case Name	**Viewing Messages**
Actors	User
Preconditions	● User is logged in and is either the recipient of the message or the sender of the message
Goal	Display a message to the user who was either the recipient or the sender

Scenario	1. User logs in 2. User clicks on a chat and is shown a list of all of the messages with timestamps within the chat
Exceptions	

Use Case Name	**Deleting Messages**
Actors	User
Preconditions	● A chat/message exists
Goal	User can delete any messages they have sent or received
Scenario	1. User logs in 2. User opens a chat 3. User clicks the “Delete chat” button 4. The messages are removed from the interface
Exceptions	● Database fails for any reason

Use Case Name	**Creating a New Group Chat**
Actors	User
Preconditions	● User is logged in and clicks the “create new chat” button
Goal	To create a new chat with two or more people including the user
Scenario	1. User logs in 2. User clicks the “create new chat” button 3. User adds a user to the group to message 4. User clicks “confirm new chat” button
Exceptions	● The group chat doesn't currently exist in the user group chat history log

Use Case Name	**Inviting a User to a Group Chat**
Actors	User
Preconditions	● The group chat that the user is wanting to add another user to is already created
Goal	User is able to add another user to a selected group chat so that they

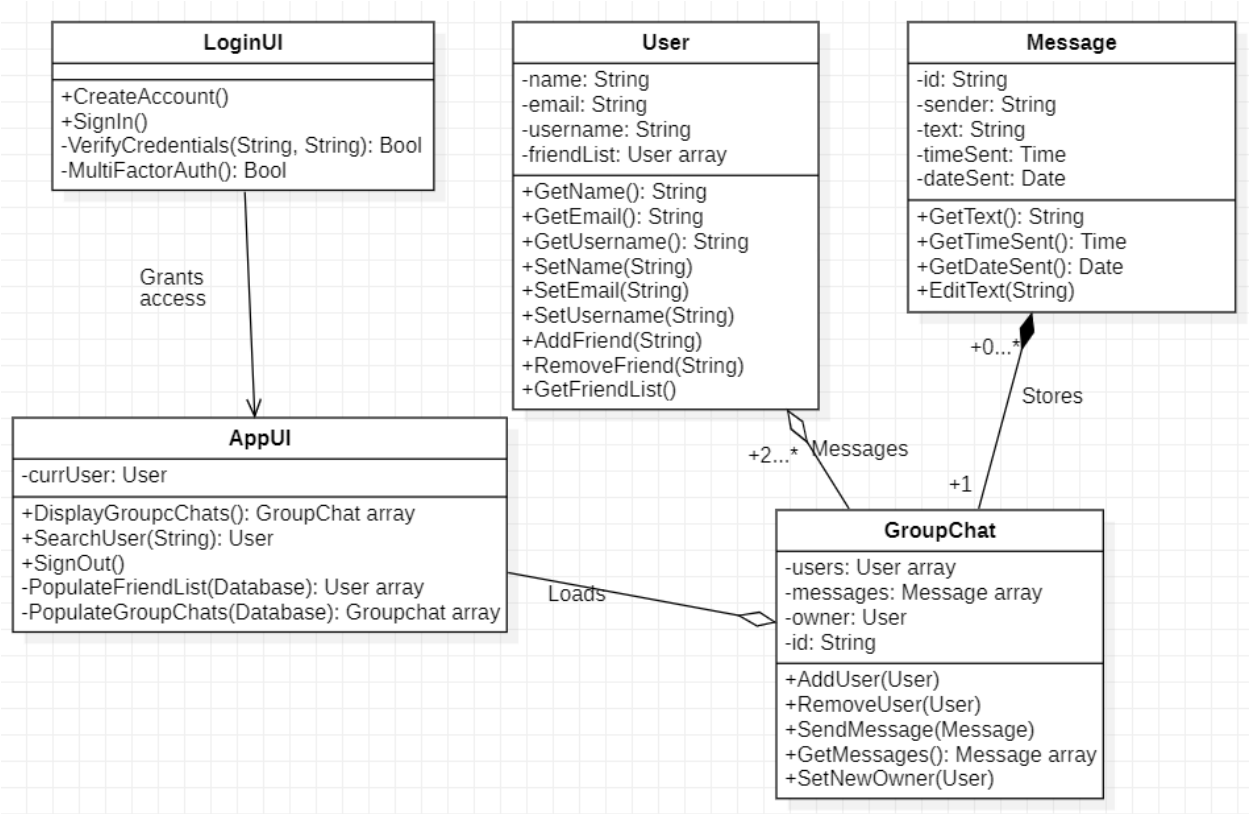
	can communicate with that user as well
Scenario	1. User selects the group chat from the interface 2. User clicks the “invite user” button 3. User enters the unique username of the user desired to be added to the group chat 4. User confirms their selection by clicking the “done” button 5. The new user is sent a request to be added to the chat
Exceptions	● The user is already in the group chat ● The username/user account doesn’t exist

Use Case Name	**Removing Yourself From a Group Chat**
Actors	User
Preconditions	● The user is apart of the group chat that they wish to leave
Goal	Allow the user to remove themselves from a group chat that they are apart of
Scenario	1. User selects the group chat from the interface 2. User selects the “options” button 3. User selects the “leave group chat” button 4. User confirms their selection 5. Interface is updated to show that the group chat is no longer visible
Exceptions	● Database fails to update user list for the desired group chat

Use Case Name	**Accepting/Denying a Request to Join a Group Chat**
Actors	User
Preconditions	● User is invited to a group chat ● User is logged into the website
Goal	To have the user accept/deny a request to join a group chat
Scenario	1. User clicks on the “invites” button 2. User clicks “accept/deny” on the request to join a group chat 3. Both choices result in the invite being removed from the users’ invite list 4. If user clicks accept the group chat is added to the user’s chat log 5. If user clicks deny the user’s chat log remains the same

Exceptions	

UML Class Diagrams:



Quality Plan:

- **Security goals:**
  - **Confidentiality**
    - When messages are deleted, they are only deleted for the party that wants to delete them (not all parties)
    - Account names are unique to avoid messages going to unwanted users
    - End to end encryption
    - Multi factor authentication for login
    - Users added to a group chat after creation shouldn't have access to previous messages.
    - Only the account owner should be able to view messages sent to or from the account
  - **Integrity**

        - Message text should not be modifiable by a third party or anyone other than the user that sent them.
            - For an edited message it should be marked as “Edited” in the UI for the application.
        - Other message aspects, such as time sent, sender, etc, should not be editable by any source (not even the user).
        - Users shouldn’t be able to be simultaneously logged into multiple accounts.
    - **Availability**
        - The application should be able to resist denial of service attacks and be always available to the user.
        - The application should not provide the user with any frustrations with regards to logging in and making use of its services (i.e. security shouldn’t come at the cost of the user’s ease of use).
- **Security metrics:**
    - **Confidentiality**
        - The percentage of how many cases where the messages that got sent are viewed by another person who is not the sender or receiver should be 0%.
    - **Integrity**
        - The percentage of how many cases where the messages have their body modified due to malicious attack, software or hardware failures after being sent by the sender should be 0%.
    - **Availability**
        - The percentage of messages that are lost due to system failures should be 0%
        - The percentage of messages that are deleted by a user who is not the sender or receiver should be 0%