Project Documentation

End to End Encrypted Chat Program Report

Ali Jalil, Josh Egwaikhide, Nick Chalardsoontornvatee

GitHub Repo Link: https://github.com/A-J21/Java-Encrypted-Messenger

1. Scope and Requirements

In the misinformation age, technology has reached a point where the average user regularly has their trust shaken by the tech and institutions meant to enhance their daily lives. Between data breeches and large companies selling user data to the highest bidder, consumers feel an increasing need for privacy policies and tooling that upholds and promotes those policies. The most basic of these tools is a private means of communication, an end-to-end encrypted chat application.

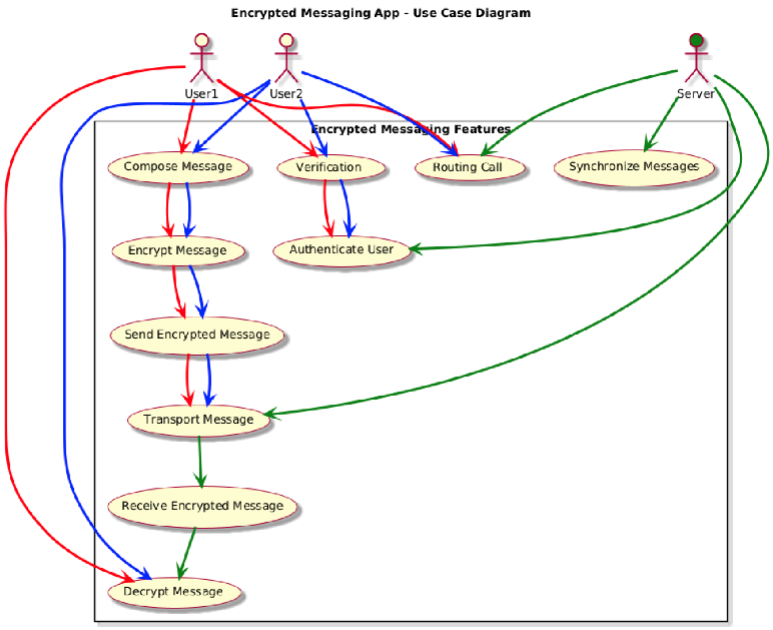
Functional Requirements:

- End-to-end encryption: the application must encrypt and decrypt messages to secure the contents of a message while in transit.
- Authentication: the application must authenticate and confirm user identities.
- Error handling: the application must handle errors such as connection issues and conditions where messages fail to send or arrive.

Non-functional Requirements:

- Security: the application must ensure confidentiality, integrity, and authenticity of the users and messages they send.
- Usability: the application must be user-friendly, especially for users that tend to not be tech-savvy.
- Reliability: the application must be able to recover from failures and errors.
- Availability: the application must be available to users for the expected service duration and uptime.

2. Use Case Diagram



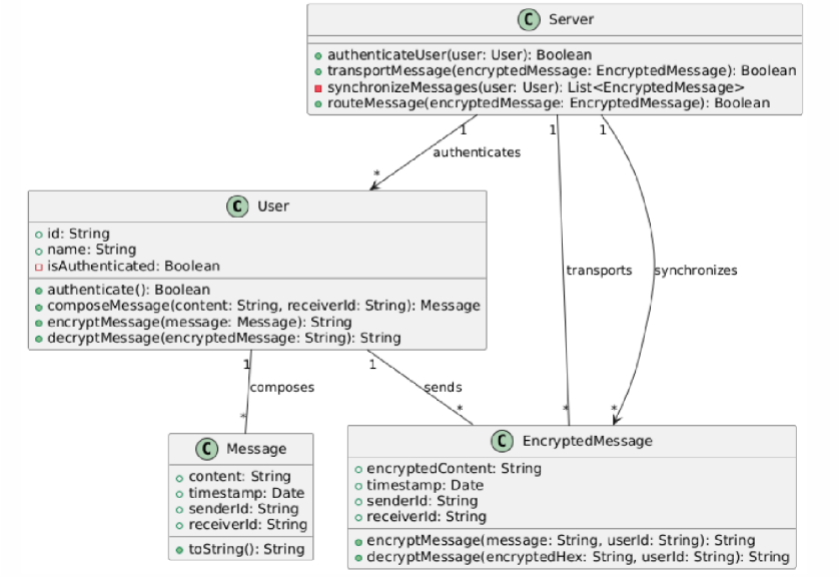
3. Use Case Document

1. Define Actors:
   - Identify the different actors (users or external systems) that will interact with your project.
   - User 1 and User 2 and Server
2. List Use Cases:
   - Based on the actors you identified, outline the key functionalities of the system.
     - User 1 and user 2 would have the same use case
       1. Verification
       2. Encrypted Message
       3. Decrypted Message
       4. Broadcast
     - Server Use Cases
       - Authentication
       - Broadcast
   - Each use case should describe an interaction between an actor and the system.
     - User 1 and 2 would verify and then authenticate with the server (Sign/log in)
     - User 1 or 2 composes a message, the message is encrypted on the user's client and sent to the server. Server broadcasts the encrypted message to the other user. The other user retrieves the message from the server and decrypts the message on their client code.

4. Design Class Diagrams

Milestone 2 Class Diagram

Group Members: Ali Jalil, Joshua Egwaikhide, Nick Chalardsoontornvatee



5. Class Diagram Document

Class Descriptions:

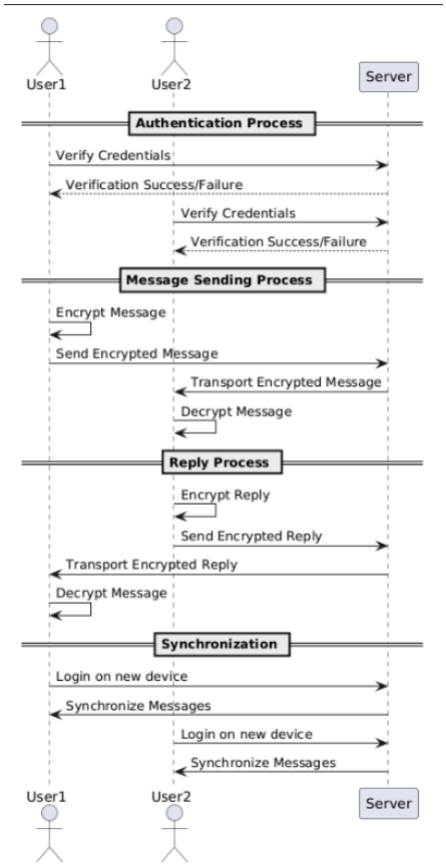
1. User:
   - Represents a participant in the messaging system. It has attributes like id, name, and isAuthenticated, which tracks the authentication status. Methods allow the user to compose, encrypt, decrypt, and authenticate messages.
2. Message:
   - Represents a message created by the user, with attributes such as content, timestamp, senderId, and receiverId. This class acts as a container for messages in plain text, stored on the user's client.
3. EncryptedMessage:
   - Holds encrypted content, along with timestamps and sender/receiver IDs. Like Message, it serves as a data container. Unlike Message, this class represents messages in their encrypted form and is transferred to the server for routing to the intended recipient.
4. Server:
   - Manages user authentication and message transport. It has a collection of active users and methods like authenticateUser, transportMessage, and routeMessage. The server also includes a private method, which ensures that messages are consistent across users and that all sent messages are received by the correct user upon their next login.
   Relationships Summary:
5. User and Message/EncryptedMessage:
   - Users create and interact with Message and EncryptedMessage objects. They use these classes to compose messages, address them to specific recipients, and then encrypt the message to ensure that only authorized users can access its content.
6. Server and User/EncryptedMessage:
   - The server handles user authentication and message routing, but it does not own the users. It facilitates communication between users by transporting EncryptedMessage objects. The server ensures the delivery of

messages to the correct user or retains them until the intended user logs in again.

6. Application Implementation

See Source Code

7. Sequence Diagrams



8. Documented Code

See Source Code