Final Report

Part 6

Team: Conor Parrish
Cathal Killeen
Daniel Yedidovich

Title: AFCA Chat

Project Summary: A simple chat application with a desktop interface which allows for text and multimedia communication with a goal for decentralized end-to-end messaging. Once delivered, encrypted messages are deleted from central server.

Features and Respective Level of Implementation:

User Requirements			
ID	Requirement	Completed?	Notes
US-001	As a user, I need to be able to send and receive messages to/from any other user	Yes	
US-002	As a user, I need to be able to sign up for the application application - create unique account with email and password	Yes	Limited functionality. Account not persisted if server shuts down
US-003	As a user, I need to be able to search for other users	No	
US-004	As a user, I need to be able to customize my profile	No	
US-005	As a user, I need to be able to login to my account using my username and password from any device	Yes	Unique user ID is maintained on server
US-006	As a system admin, I need to be able to view server storage data	No	
US-007	As a user, I need to be able to reset	No	

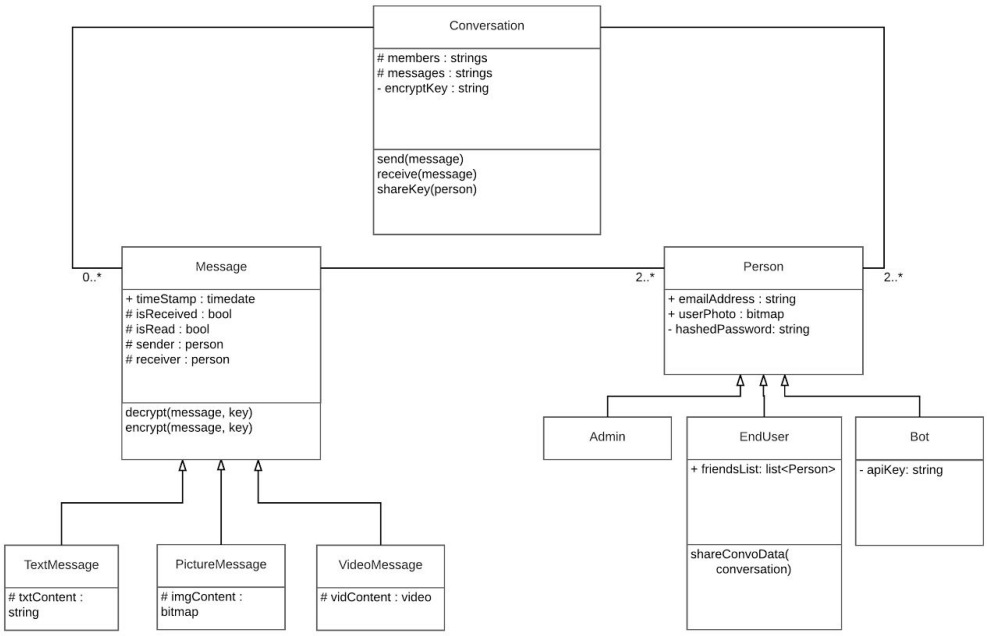
	my password with a simple UI		
US-008	As a system admin, I need to be able to send password reset links	No	
US-009	As a user, I need to be able to access and send media from my device	Yes	
US-010	As a user, I need to be able to create a group chat	Yes	
US-011	As a bot, I need to be able to be available for users to message and interact with	No	Bots not implemented
US-012	As a user, I need to be able to see when the recipient receives/reads their messages	No	Read receipts not implemented. However, user notified if another user goes offline
US-013	As a third-party developer, I need to have access to an API to create bots that can be used by end-users	No	Bots not implemented
US-014	As a user, I need to be able to customize the app	No	GUI not main focus
US-015	As a user, I must be able to edit basic info on my account (change username, email, passsword etc)	50%	Username can be changed

Non-Functional Requirements			
ID	Requirement	Completed?	Notes
NFR-001	Platform Constraints: Messages must be end-to-end encrypted	No	
NFR-002	Platform Constraints: Users must be able to recover messages (transfer messages from old device to new device)	No	
NFR-003	Platform Constraints: Received messages should be deleted from the server	Yes	
NFR-004	Platform Constraints: Server must be able to detect when a message has been received by an end user	Yes	Via socket

NFR-005	Performance: Encryption and decryption must take no more than 2 sec	No	Encryption not implemented
NFR-006	Platform Constraints: Admins should not have access to messages sent by users	Yes	

Class Diagrams:

Part 2 Class Diagram:

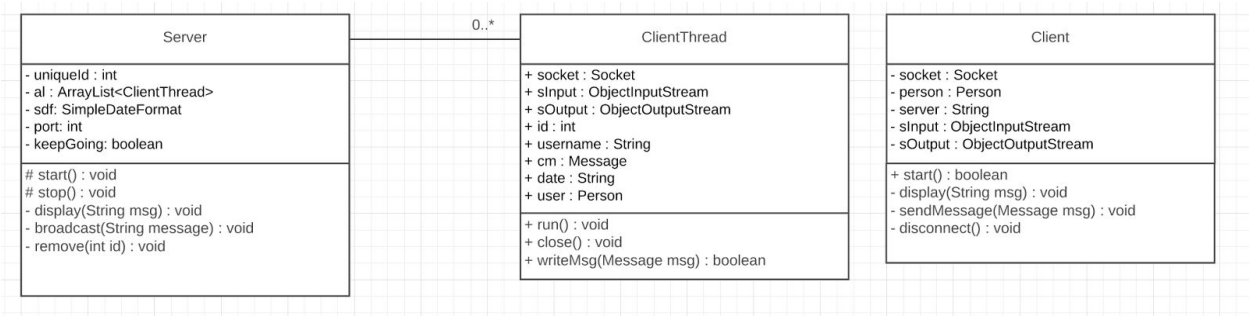


Part 6 Class Diagram:

As you can see, our class diagram changed a lot between part 2 and the final version. The main thing that changed was that we needed to implement a different thread for each client and create a server class to handle sending and receiving messages for each user. This was a fundamental change in the structure of our project, which is reflected in the modified diagram. Additionally, the conversation class was no longer necessary, since the server and client threads do all of the work coordinating the connections between the clients and storing and sending the messages. We also were not able to implement image or video messages so those classes were omitted from the class diagram, but BehindTheScenesMessage was added so that we could send logout and other such backend communication.

Design Patterns:

The structure of our prototype is Client/Server. This is implemented through a modified Observer design pattern. The clients are the observers. When a message is published to the server, the respective clients are notified instantly. See below for the class diagram of this implementation.



What We Have Learned:

We learned a lot about the process of designing a system during this project. Creating the class diagrams and use cases really helped us to formalize how our system would work, which was especially important for working together to build it as a team. However, we also realized during our implementation that some of our assumptions about the structure were wrong or incomplete. We also learned to begin with a small scope and add stretch functionality as time allows. Since we initially imagined more features than we had time to implement, we ended up eliminating some functionality from our design. We believe this highlighted the importance of iteration, and how the design, analysis and creation of the system were very interlinked. We chose to focus on the backend of our system, rather than getting too obsessed with the GUI, which none of us are very experienced with in Java. When developing our project, we noticed the power of low coupling when adding new features. Adding a new feature didn't require a rewrite of the entire system, usually only one or two classes needed to be updated to add a new feature.