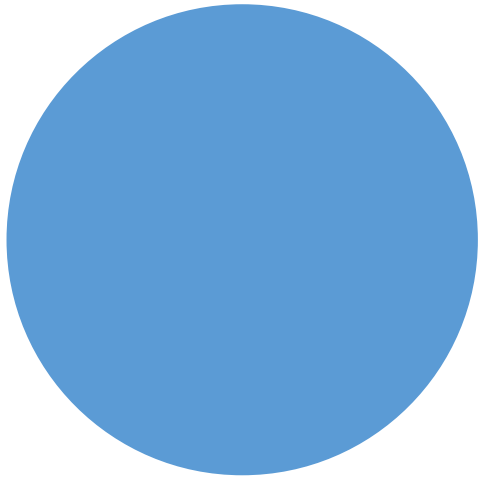
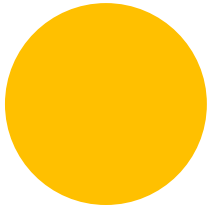
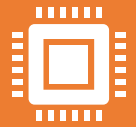



Prattle – A chat messaging system

CS5500 Spring 2020, Section 5

Matthew Sobkowski, Ben Steele, Jeremiah Holbrook, Evgenii Shatokhin

Project Overview



CCIS has acquired a new communications server

The server only carries the basic messaging capabilities

The server shall be extended to provide advanced messaging capabilities, slick UI design, and scalability for high loads.

Customer focused

Project Requirements

- What is the customer looking for? How do we distinguish from other applications?

- Core Project requirements:
  - The server must be written in Java
  - The server must extend the current system

Customer Requirements

- Customer requirements:
  - Privacy/Security
  - Encryption
  - Scalable
  - Basic communications functionality – Sending and receiving messages
  - Users can create groups
  - Groups support moderator functionality, inviting and removing users, creating and deleting groups
  - Government Compliant
  - Social Aspect

Team 4

- The team consists of:
  - Matthew Sobkowski
  - Ben Steele
  - Jeremiah Holbrook
  - Evgenii Shatokhin

The Final Product

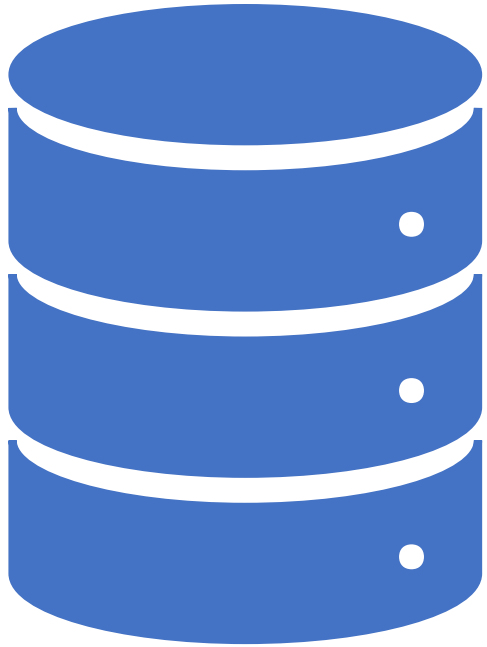
System Functionality

- The current system supports the follwing functionality:
  - User login, including registering as a new user
  - Password protection
  - Direct and group messaging
  - Managing groups
    - Creating groups
    - Moderators
    - Inviting and removing users
  - User Status
  - End-to-end encryption
  - User settings (Translation, Filter)
  - Account management (password change)
  - Full backend implementation. All data stored.

Completed Requirements

Login

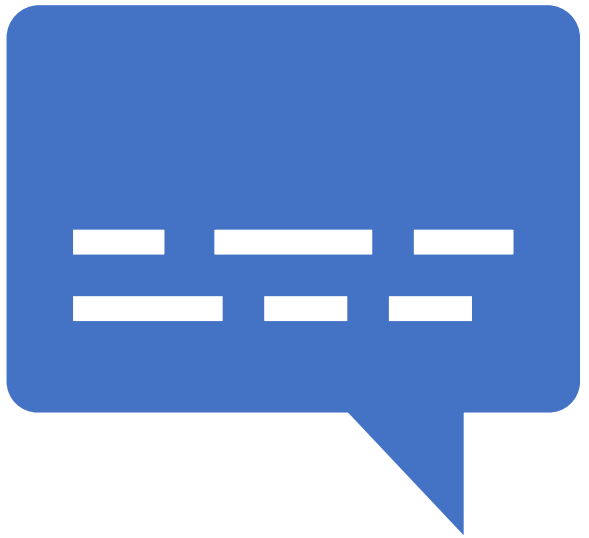
- The system must support unique user log in information. A user will register a username and password, which the system will store and keep as persistent data.
- The system may support password restrictions. This could include capitalization, unique character requirements, or minimum character length.
- The system will encrypt the user's login information when they create a new account and will store it encrypted while at rest.
- The system will encrypt a user's login information when they are attempting to login and will query the encrypted data against the database. The system will never see the user login information unencrypted.



Completed Requirements

Messaging

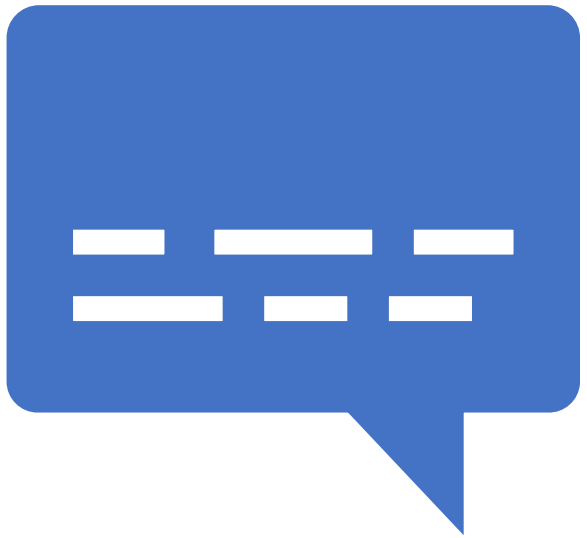
- Users must have the ability to send messages to channels. This will include private messages, group channels, feeds, etc. The user will create a new message client side, and then will send the message to the server.
- All messages must be sent to a specific channel.
- When a user sends a message to a channel, the system will push the message to all active users in that channel.
- The system will store all messages that a user sends to it. Stored messages will be able to be recalled later.
- The system will encrypt all messages received from clients and transmit them encrypted to the receiving clients.
- All messages will have a timestamp.



Completed Requirements

Messaging

- Messages will be delivered/rendered in the order they were sent. This is determined by the timestamp assigned to the message.



Completed Requirements

Channel

- Server treats channels as a function of groups
- Messages travel through channels to specific channelIDs that represent either users or groups.
- Server-side implementation of channel interface will be implemented via group-wide channel, DM channel and wall channel

Completed Requirements

Member

- Users should be able to use “personality” icons for their profiles.
- Users should be able to set availability (Do not Disturb, Online, Unavailable)

Completed Requirements

Groups

- Users can be part of groups.
- Groups must have at least one user, and at least one moderator
- Moderators can kill groups, and can dictate who can be invited and rejected from joining
- Groups can have more than one moderator.
- Moderators can transfer powers to other users in a group.

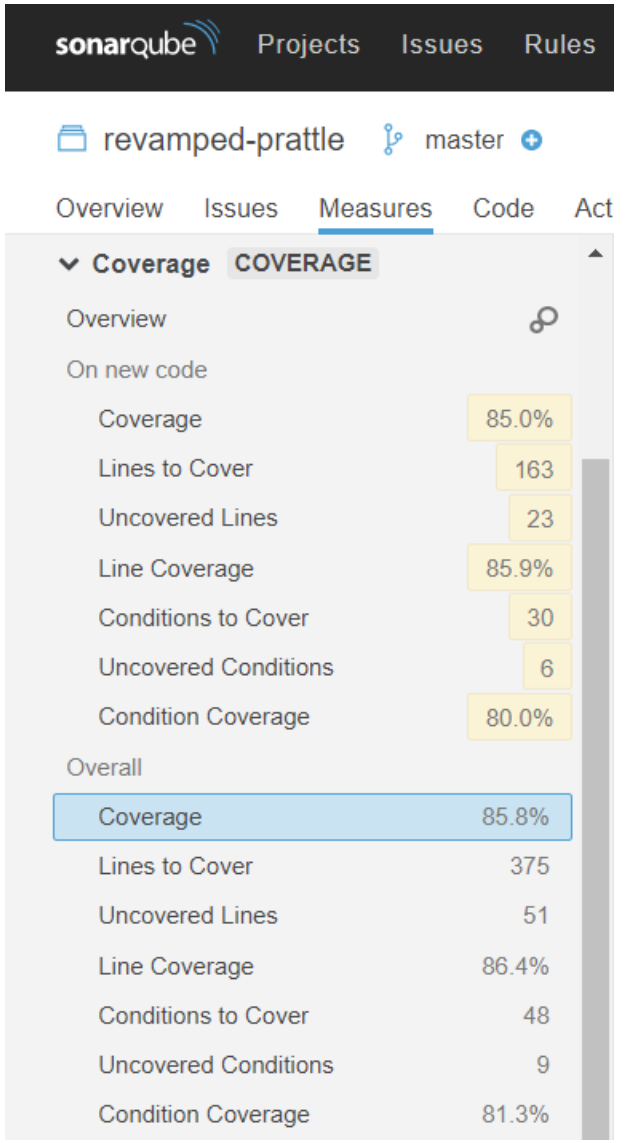
Functionality Assessment

- Overall, the team did a fantastic job completing the requirements that were set at the beginning of the project.
- Core functionality completely integrated. System is developed in a way that leaves it easily open for extension/modification

- The current system provides a functional, efficient, and secure communications system for the client. The system is scalable for large scale operations

Statistical Analysis

- Required to hit 85% Line coverage, and 50% conditional coverage
- Sonarqube shows 86.4% line coverage, 81.3% conditional coverage
- Overall 85.8% coverage



Development Process

- Development for each sprint followed roughly the same process
  - Determine sprint goals at the beginning of the sprint, taken from the SRS, Trello, etc.
  - Assign sprint goals to group members
  - Discuss integration process
    - Multiple members touching same part of the code base
    - Certain systems required to be complete for others to properyl function
    - Ensuring no merge conflicts, smooth integration, etc.
  - Week of development, followed by a group meeting haflway through sprint to discuss design, development, any roadblocks, etc.
  - All code pushed by Wednesday night/mid day thursday.
  - Sprint meeting friday afternoon

Looking Back

- The development process worked well. Allowed for constant communication with team members while allowing individual development and design
- Improvements could have been made to include more meetings, possibly more daily communications. But this was not a hinderance to the overall project.

- Used various forms of communication, including slack, text messaging, discord. Discord was main platform, used for all meetings as well.

Team Dynamic

Team members worked well together

Built upon team members strengths for development and roles

Communication helped in identifying any issues and ensuring the proper help was given to overcome roadblocks

Transitioning to System the client

THE CURRENT SYSTEM PROVIDES ALL THE CORE FUNCTIONALITY OF A FUNCTIONING CHAT SYSTEM AS REQUESTED BY THE CLIENT

ADDITIONALLY, THE SYSTEM PROVIDES A ROBUST AND FLEXIBLE SECURE ENVIRONMENT. END-TO-END ENCRYPTION AND ADDITIONALLY DATA AT REST ENCRYPTION ENSURES SECURITY FOR THE USER.

SIMPLE, USER FRIENDLY UI THAT IS EASY TO USE AND GET TO KNOW

Looking ahead

System is developed in a modern environment, allows for extensibility for future development

- Javascript, HTML front end
- Java Server Side
- AWS, DynamoDB backend for data storage

Next steps would be to implement more customer specific requirements

- Increased user security
- Third party login
- Social Aspects