MSD Team 102
Slick Messaging

Avik Sengupta
Tanmay Sinha
Cole Clark
Sarah Lichtman

- System Functionality
- Job Quality
- Process and Teamwork
- Technology Transfer

Our goals

- Enable two or more users to communicate with each other via text messages
- Host on the server on AWS cloud and ensure it is running 24x7 to facilitate client requests
- Achieve text communication between users and groups via private and group messages
- Add functionalities to recall messages not yet received, search for a message, flag inappropriate messages, etc

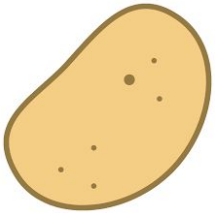
System Functionality

- Sprint 1:
  - Getting the legacy code base running (Broadcast messaging)
- Sprint 2:
  - Adding a notion of user and groups of users to the system
  - Directing messages to individuals and groups
  - Add message persistence to messages sent to individuals and groups
- Sprint 3:
  - Queue messages
  - Duplicating (dup'ing) a message targeting a specific user or group
  - Recall messages sent
  - Wrap message stored in the system with the sender's and receiver's IP address
  - Add a parental control feature
  - Search for a message

Is the chat server useful?

- The chat server in its current state is good for basic text communication
- The feature to enable parental controls is not found in many apps without any third party support
- Support for sending images, audio or video would not add any non existing functionality already found in other messaging applications
- Perfect for communication among the developers as it guarantees privacy
- Privacy is not guaranteed for any other user as we have message duplicating feature in place

Our competition



Our app in current state

Jira Evidence

- Used smart commits to update progress made with tasks in JIRA
- Each requirement was added as a task in JIRA
- Tasks were subdivided into smaller tasks
- Tasks were assigned to team members
- Ticket numbers for task were reference when using git

Sarah Lichtman changed the status to Done on ~~MSD102-54~~ - Dynamically turn on or off logging in the system

Tanmay Sinha changed the status to Done on ~~MSD102-47~~ - Queue messages for offline users, send when onlin

Cole Clark changed the status to Done on ~~MSD102-66~~ - Stretch 9: search for message

Avik Sengupta changed the status to Done on ~~MSD102-53~~ - Dup messages for wiretaps

- System Functionality
- Job Quality
- Process and Teamwork
- Technology Transfer

Test Coverage and Code Quality

- Testing
  - Tested all Prattle functionality (new and legacy) to ensure the expected behavior of the program
  - Met the expectation of 85% branch coverage (85.1%)
  - Exceeded the expectation of 50% condition coverage (78.0%)
- Code Quality
  - Duplication of lines stayed under the allowed 3% (2.5%)
  - No bugs or vulnerabilities detected by SonarQube
  - Minimized major code smells

team-102-F18 Passed

0 A	0 A	27 A	85.1%	2.5%
Bugs	Vulnerabilities	Code Smells	Coverage	Duplications

Overall	
Coverage	85.1%
Lines to Cover	1,522
Uncovered Lines	179
Line Coverage	88.2%
Conditions to Cover	676
Uncovered Conditions	149
Condition Coverage	78.0%
Tests	
Unit Tests	76
Errors	0
Failures	0
Skipped	0
Success	100%
Duration	6min

Code Maintainability

- Documentation
  - Thorough javadoc comments for classes, methods, and constants to increase readability and maintainability
  - Followed standard naming conventions for functions, variables, and constant values

```java
/**
 * Construct a string containing information about the requested message from the archive
 *
 * @param sentTime     SQL TIMESTAMP of when the message was received by the server
 * @param messageType  Three-character string indicating the type of the original message
 * @param fromUser     Username that the archived message was sent by
 * @param toUser       Username that the archived message was sent to
 * @param text         Text in the archived message
 * @return A string describing the archived message
 */
private static String constructArchiveText(String sentTime, String messageType, String fromUser,
                                           String toUser, String text) {
  String result = "time: " + sentTime;
  result += " type: " + messageType;
  result += " from: " + fromUser;
  result += " to: " + toUser;
  result += " text: " + text;
  return result;
}
```

Performance Over Time

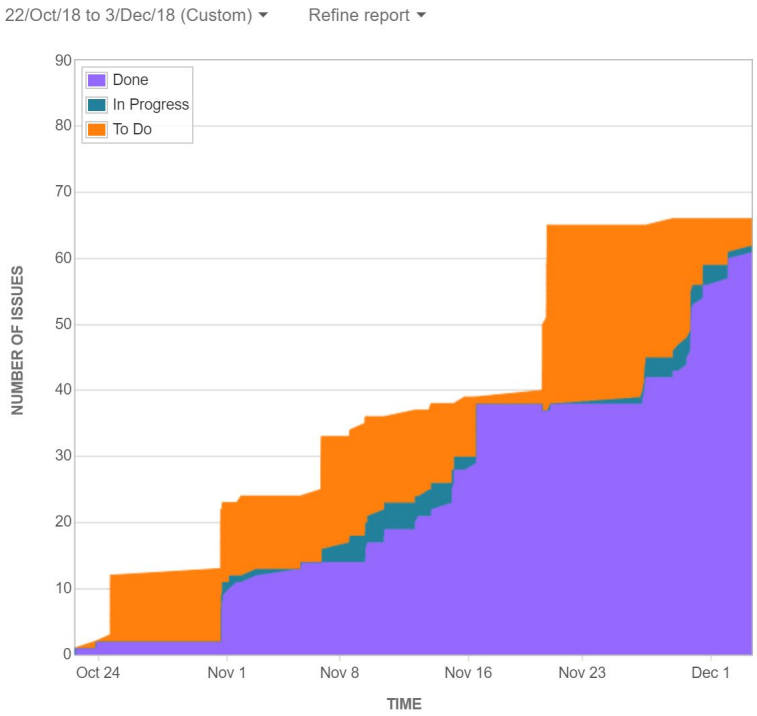
- Completed all base expectations in all sprints
- Added more system functionality through completion of stretch goals over the course of the project
- Notable stretch goals completed: password encryption, message persistence using MySQL database, search messages, parental controls, and recall messages

	Base Expectations		Stretches		
Sprint #	Completed	Total Assigned	Small	Medium	Large
1	7	7	3	0	0
2	4	4	0	1	1
3	5	5	3	1	1

- System Functionality
- Job Quality
- Process and Teamwork
- Technology Transfer

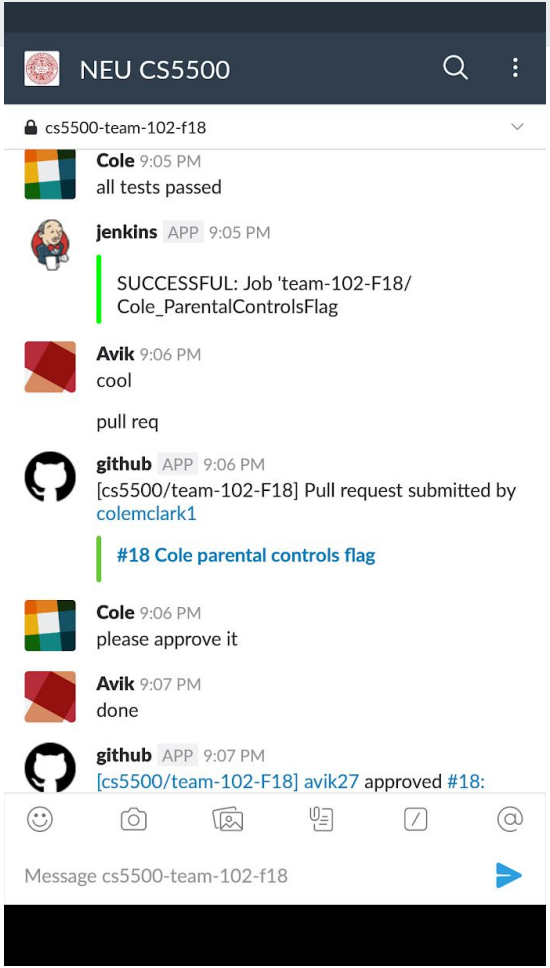
Process

- Automating the process
  - Build and test via Jenkins on Alpine Docker, with setup provided by Alex Grob
  - Deploy master via Jenkins to an AWS server
  - Promote via Slack integration with Github and Jenkins
- Process organization
  - Task backlog organized and refined in Jira
  - “Smart commits” used to link tasks to Github commits



Teamwork

- Working as a Team
  - In-person meetings at least once per week, for task setup and feature merging
  - Slack communication as needed, usually daily
  - Work completed in timely manner, with reasonable and mutually-agreed-upon division of labor
- Challenges and Solutions
  - Difficulty scheduling in-person meetings
    => remote work with collaboration over Slack
  - Task distribution issues in Sprint 2
    => collaborating on Jira task entry for Sprint 3



- System Functionality
- Job Quality
- Process and Teamwork
- Technology Transfer

Technology Transfer

- AWS
  - Application is deployed in a running AWS instance
  - Can be deployed in any AWS server by modifying the Jenkins File
  - Using a MySQL Database to handle multiple users, groups and messages
- Testing
  - The application is well tested using JUnit test cases
- Documentation
  - The application is well documented using JavaDocs for easy understanding of the code
- Best Practices
  - Using git to store the codebase
  - Using Jenkins, Slack and Jira for continuous integration and task allocation
  - Singleton pattern for initializing database connection
  - Properties file for database, log4j and forbidden words

Path Forward

- As a product:
  - MIME type messages
  - Make the Prattle Server robust to scaling and heavy loads
  - Assess the Prattle Server using JMeter
  - Develop standout Chatter client to differentiate from existing products
- As a project:
  - Provide socket use and testing examples
  - Provide Mockito examples and test cases