

# Managing Software Development Project Presentation

TEAM-203

Akshay Sadarangani

Chetan Mahale

Nipun Midha

Peter Mascarenhas

# Acknowledgements

We would like to thank the following people for making this project possible:

- Professor Mike Weintraub under whose guidance we developed this project
- Matthew Hertz and the instructors for the initial code base
- CS5500 Fall 2018 TAs for reviewing our work throughout the semester
- CCIS course staff for helping us set up our environments
- Professor Virgil Pavlu for lending us his camera
- Our friends and family

# Introduction

Slick is the next Slack!

Slick is a socket based server that has the capabilities of sending messages to users and groups of users at scale. With features like parental control, recalling sent messages, private messages, group messages and broadcast messages this application is ready to be in use.

The bundled client application a.k.a. Chatter helps users connect to the server and use it with ease. Multiple users can connect to the server at once due to its robust nature and carry out activities with low latency.

# Technical Specifications

Language	Java version 1.8
Testing framework	Junit 5
Version control	Github (github.ccs.neu.edu)
Issue tracking	Jira
CI/CD	Jenkins
Messaging/Alerts	Slack
Cloud Environment	AWS

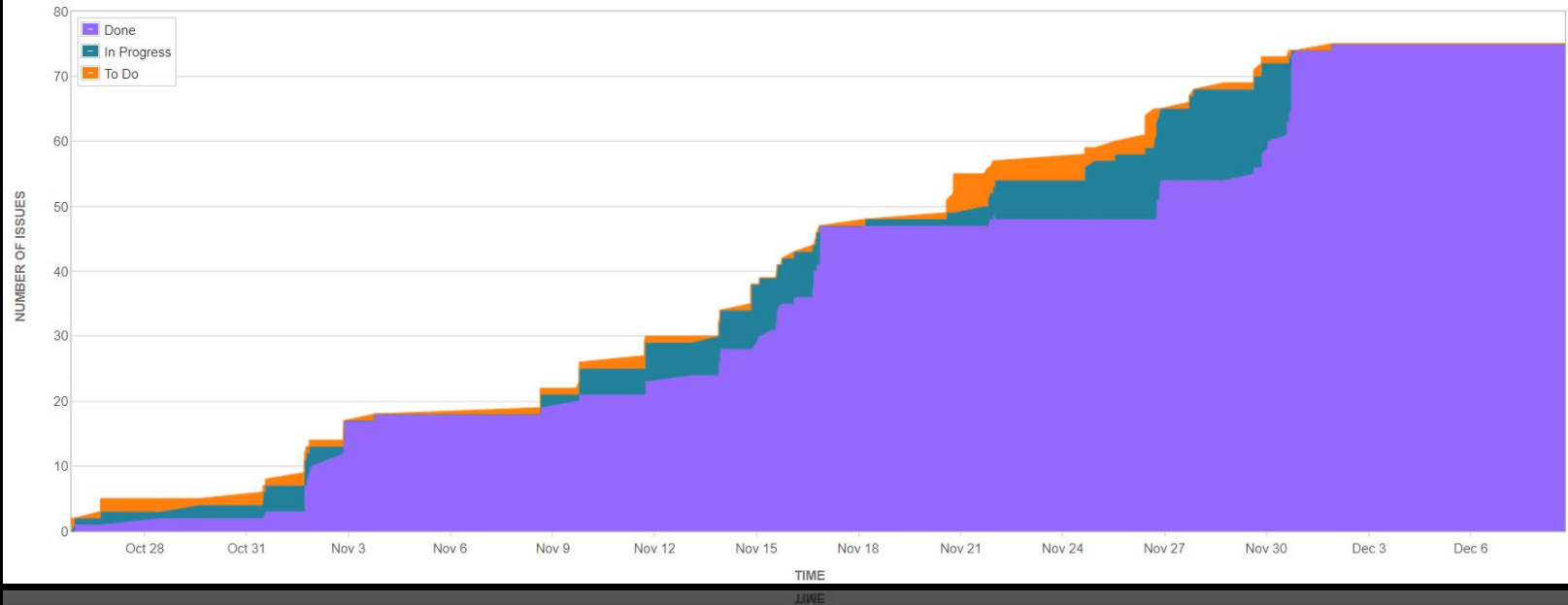


# Planning - Jira

MSD203 board

Cumulative Flow Diagram [Switch report](#)

25/Oct/18 to 8/Dec/18 (Custom) [Refine report](#)



# Continuous Integration - Jenkins



team-203-F18

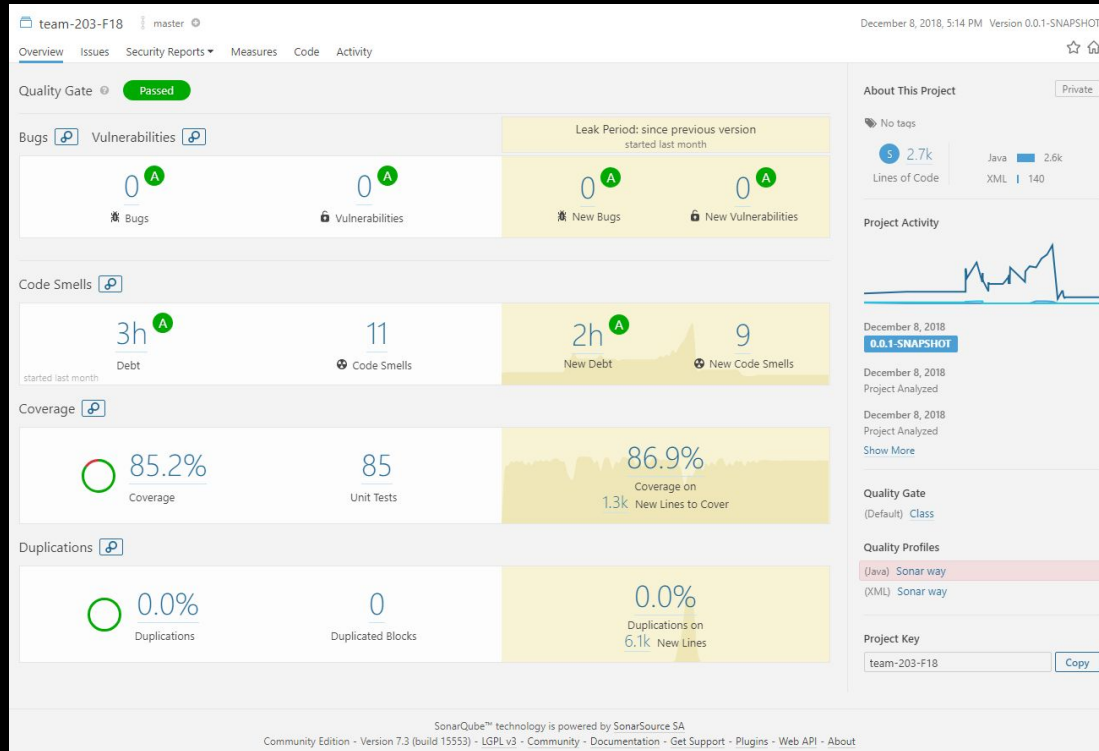
team repo for team-203-F18

Branches (28)







Pull Requests (54)

S	W	Name	Last Success ↑	Last Failure	Last Duration
		master	1 hr 16 min - #25	21 days - #14	1 min 32 sec
		KingSingleton	1 hr 23 min - #85	1 hr 33 min - #84	4 min 38 sec
		KingSingleton	1 hr 33 min - #82	1 hr 33 min - #84	4 min 38 sec
		master	1 hr 16 min - #25	21 days - #14	1 min 32 sec

# Continuous Testing - SonarQube



# Continuous Deployment - AWS

Filter by tags and attributes or search by keyword											
<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	
<input type="checkbox"/>	Prattle NEW	i-0aacde26618ce4f90	t2.micro	us-west-2a	 running	 2/2 checks ...	None	 ec2-35-166-190-64.us-...	35.166.190.64	-	
<input type="checkbox"/>	Prattle NEW	i-099c0650918ce4f90	t2.micro	us-west-2a	 running	 2/2 checks ...	None	 ec2-35-166-190-64.us-...	35.166.190.64	-	



# Slack Integrations



**github** APP 5:01 PM

[cs5500/team-203-F18] Pull request closed: [#35 King singleton](#) by 4kshay

[[team-203-F18:KingSingleton](#)] 1 new commit by 4kshay:

| [e34c980](#) Update README.md - 4kshay



**jenkins** APP 5:11 PM

SUCCESSFUL: Job 'team-203-F18/master



**Slick Bot** APP 4:52 PM

Something went wrong during data transfer @ Slick



**Slick Bot** APP 10:34 PM

Invalid Login Attempt @ Slick

# SPRINT 1

# Overview

## Functionality

- Setting up codebase, Jenkins, Slack, Jira
- Running the codebase and connecting a Database
- CRUD for Users and Groups
- Sending messages individuals and groups

## Environment

- Use Junit 5 and java version of 1.8
- Deploy it to AWS
- Inform failures on slack/email
- Smart commits in git

# Initial Design

- Write test cases and ensure 85% coverage of the codebase
- Split modules amongst ourselves
- Setup MongoDB
- CRUD for User and Group

# Development Phase

- 85% testing coverage was not easy.
- Clearing code smells, refactoring test cases
- User creation and messaging (broadcast only)
- Notion of groups was incomplete due to lack of time
- Integrating with Slack
- Deployed to AWS (Manually)
- Create the habit of using Smart commits

## Challenges

- Understanding the codebase
- We underestimated the time it takes to achieve 85% test coverage and create a good test suite.

## Victories

- MongoDB was easy to setup and manipulate
- Integrations with Slack and AWS went smoothly

# Result

- Understood the codebase in depth while writing test cases
- Completed goals and achieved test coverage
- Familiarity of
  - common code smells
  - git smart commits
- Incomplete Group CRUD and messaging

# Sprint 1 Major Achievements

☆ [team-203-F18](#)

Passed

Private

Last analysis: December 8, 2018, 5:14 PM

0 A

 Bugs

0 A

 Vulnerabilities

11 A

 Code Smells

 85.2%  
Coverage

 0.0%  
Duplications

2.7k S

Java, XML

 0/0

 0/0

 0/0

 0/0

 0/0

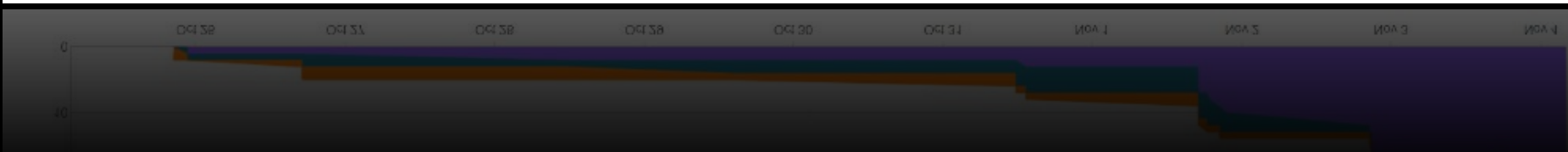
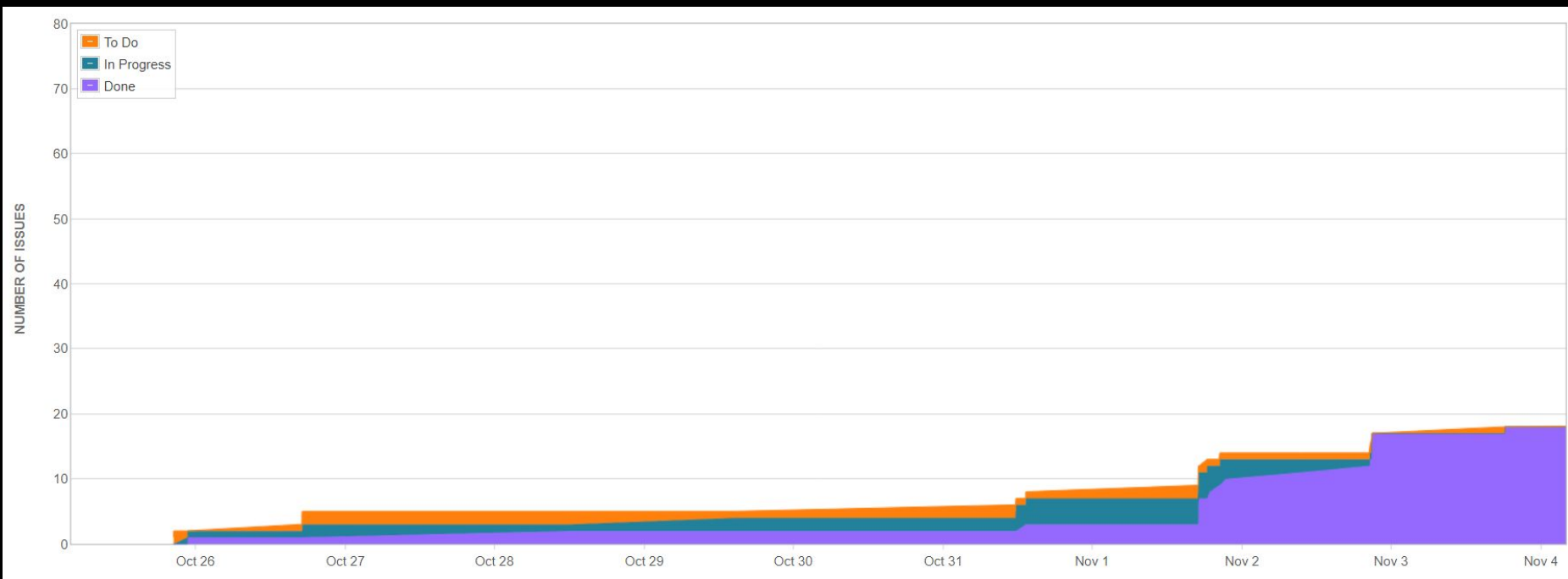
 0/0



## Learnings and Future work

- Explore tools like Mockito for a better test suite
- Find a solution to run Junit 5 tests serially
- Complete Group functionality

# Sprint Progress



# SPRINT 2

# Overview

## Functionality:

- Adding a notion of User and Groups
- Persistence of user and groups
- Login with username and password
- Directing messages to individual and group reply all
- Persistence of all the messages
- Sending messages of different MIME types

## Environment:

- Jenkins failure to be alerted by Slack
- Github should inform the team of PRs via Slack

# Initial Design

- Selected NoSQL database
- User and Group CRUD operation
- Sending private and group messages
- Message Persistence
- Retrieving persisted messages in the form of history messages

# Development phase

- Decided on MongoDB
  - Easy to set up
  - Not structured hence easy to develop against
- We divided the tasks according to layers of application
  - To achieve expertise at different application layers
  - Faster turnaround time for issues
- Test coverage was monitored on the fly

# Challenges

- Conversion of our DB object to application POJO
- Persisting the whole message object in database
- Sending a mime message of video and gifs was a big challenge
- Continuous integration on AWS required permissions
- Making chatter more user interface friendly
- Environment issues

# Victories

- Provided Help functionality
- Color coded sent and received messages

# Result

- Completed all goals for this sprint
- Extra functionality of viewing history messages on login, Color coding messages and Providing HELP functionality was welcomed
- MIME for video and gifs was timing out
- Adding new functionalities were easier

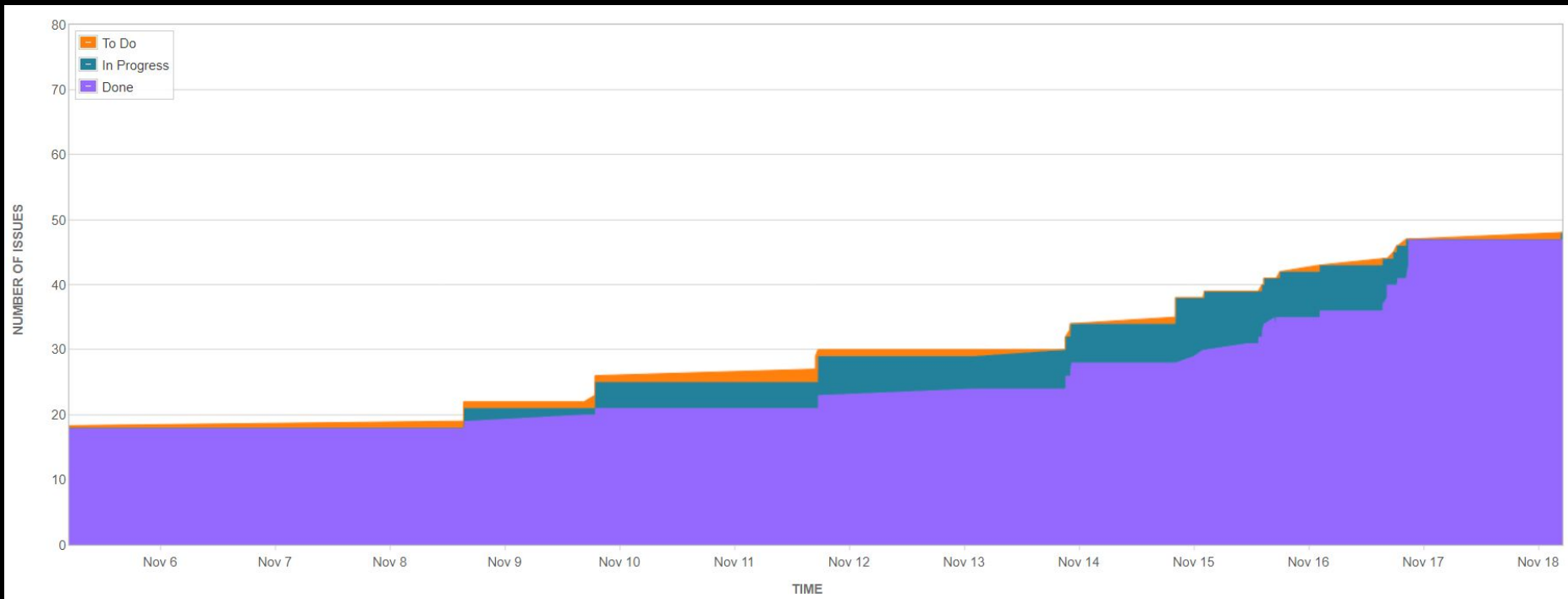


# Sprint 2 Major Achievements

```
F:\team-203-F18\Project_src\Chatter>mvn exec:java -Dexec.mainClass=edu.northeastern.ccs.im.chatter.CommandLineMain -Dexec.args="ec2-35-166-190-64.us-west-2.compute.amazonaws.com 4545"
[INFO] Scanning for projects... [INFO]
[INFO] -----
[INFO] Building Chatter 0.0.1-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Chatter ---
      ::Welcome to Slick::
Welcome! Begin by logging in using LOGIN_USER <username> <password>
Tip: Use the help command to see a list of commands at any point!
CREATE_USER prez prez
User Created
UPDATE_PASSWORD prez test
Success
JOIN_GROUP akshaygroup
Added to Group
[Private Msg] akshay: Hey Prez!
PRIVATE akshay Hey Akshay! I just joined your group. Let's talk there!
Delivered with UID -> 1544374997201
[akshay@akshaygroup] Hey Gang! Prez has joined us and now our group has expanded! Yay!
[Broadcast] akshay: Anyone who wants to discuss technology related stuff, join my group @akshaygroup /155.33.132.21:52258
```

```
[Broadcast] akshay: Anyone who wants to discuss technology related stuff, join my group @akshaygroup /155.33.132.21:52258
[akshay@akshaygroup] Hey Gang! Prez has joined us and now our group has expanded! Yay!
Delivered with UID -> 1544374997201
[Private Msg] akshay: Hey Prez!
PRIVATE akshay Hey Akshay! I just joined your group. Let's talk there!
```

# Sprint Progress



# SPRINT 3

# Overview

## Functionality

- Subpoenas was introduced where an agency could keep a tab on communication between two individuals or groups
- Queuing of messages for offline users
- Duplicating messages for an agency (wiretap)
- Recalling of the messages by sender
- A toggle for Parental control on offensive messages

## Environment

- Load testing using JMeter
- Slack alerts for security concerns
- Dynamic Logging

# Initial Design

- Persist unread messages to user and display on login
- Add Subpoenas Table, CRUD and backend services to subpoena objects
- Implement Parental control by parsing the message for vulgarity against a bag of words
- CALEA compliance changes can be easily incorporated to our messages
- Recall messages (Initially understood as recall last sent message)
- Understand JMeter

# Development Phase

- Queued messages to be saved for each user
- We padded the sender and receiver IP's for every message
- Recall functionality changed from last sent message to recall any message
- Parental control moved from the Chatter Side to Prattle (with a switch)
- Added a search functionality for messages
- We explored JMeter and ran tests on our DB

## Challenges

- Misunderstood the functionalities for Recall and Parental Control
- Achieving coverage took longer than expected

## Victories

- Queuing was part of our Sprint 2 and required few changes
- It was easier to understand and extend the code base as we were very comfortable with the codebase

# Results

- Parental control is now using an API instead of the bag of words
- We achieved all the functionalities with test coverage > 85%
- Observed our fewest number of code smells
- We explored JMeter and tested a few aspects of the project like DB



# Sprint 3 Major Achievements

# Sprint 3 Major Achievements

```
SUBPOENA 5c0a0d515f28cf2516914a87
```

```
Subpoena Channel Login Success
```

```
1544162759941 /205.201.22.108:59814 [Private Msg] s3u1: sdaksjdladkas -> s3u2 /205.201.22.108:59925
```

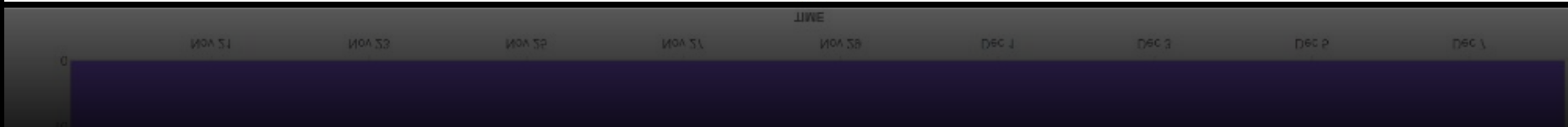
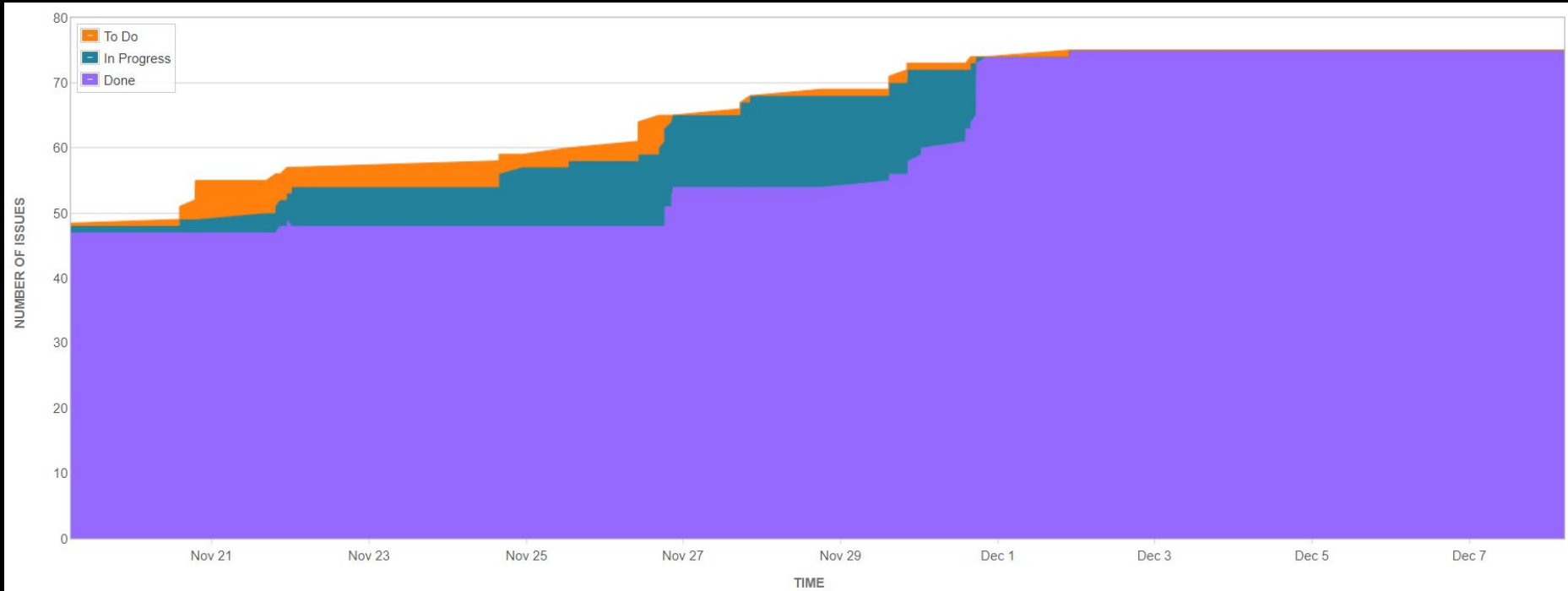
```
1544163269862 /205.201.22.108:59925 [Private Msg] s3u2: dasjdsajdlasjdldasd -> s3u1 /205.201.22.108:59814
```

```
1544163293065 /205.201.22.108:59925 [Private Msg] s3u2: kjdajdlasdlaslaslajladla -> s3u1 /Offline
```

```
1544163846085 /205.201.22.108:61459 [Private Msg] s3u2: qwert -> s3u1 /Offline
```

```
1244163846082 \502'501'55'108:0142a [Private Msg] s3u2: dmsld -> s3u1 \014111111
```

# Sprint Progress



Thank You!!