MSD TEAM 208
CHATTER INTERNET MESSENGER

HIMANSHU BUDHIA

JOHN GOODACRE

RAM PRAKASH ARIVU CHELVAN

SHASHWAT SANGHAVI

OUTLINE

- **System functionality**
- Job quality
- Process and teamwork
- Technology transfer

OUR GOALS

User-user conversation
- Search for user/conversation
- Message persistence
- User privacy
- Online users

Groups
- Create group
- Add members/admin
- Add another group

Message threads

Multimedia Support
- Images
- Videos
- Gif

Live Translation

Host the system on cloud (AWS) to provide 99%+ uptime to the users.

SYSTEM FUNCTIONALITIES

Complete feature list

1. Register and sign-in
2. Find the online users
3. Create groups and add users in it, merge two groups, make new admins for the groups
4. Conversation with individual user
5. Conversation with a group
6. Threads inside conversation
7. Audio video and gif in conversations and threads
8. Emoji
9. Live translation in 13 languages
10. Profile pictures
11. Broadcasting to all conversations
12. Private groups and users and search for them

IS THIS APPLICATION USEFUL?

- Text, multimedia and group communication
- Highly secure communication (TCP connections)
- Live translation in multiple languages (13 languages)
- Threads

- Whatsapp, slack are far ahead in the competition but we have unique features.

OUTLINE

- System functionality
- **Job Quality**
- Process and teamwork
- Technology transfer

CODE MAINTAINABILITY

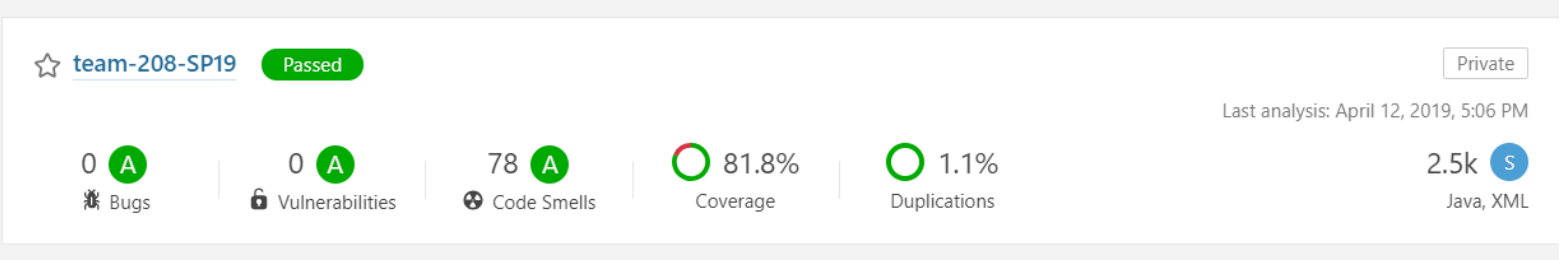
- Thorough documentation by using comments for classes and methods – increasing readability
- Followed standard naming conventions for variables, classes, and methods
- Used static constant variables for code maintainability

This would help future developers to read and understand our code easily, if this is passed on to them directly as base code.

```
import java.util.Map;

/**
 * The type Route.
 */
public class Route {

    /**
     * Get GET API response string.
     *
     * @param username the username
     * @param route    the route
     * @param params   the params JSON string with all elements as strings
     * @return the string
     */
    public static String getResponseGet(String username, String route, String params){
        List<Map<String, Object>> response;
        Map<String, Object> json = decodeJSON(params);

        try {
            switch (route) {

                case ApiMessageType.ONLINE_USERS:
                    response = ControllerFactory
                            .getUserController()
```

TEST COVERAGE / CODE QUALITY

- Test Coverage
  - We tested Chatter code functionalities to ensure expected behavior (135 tests)
  - Met the expectation for the quality gates by achieving over 80% coverage over the new and legacy code (81.8%)
  - Exceeded the expectation for code quality of 50% condition coverage (68.6%)
- Code Quality
  - Removed all the bugs and vulnerabilities
  - Ensured duplication of code is below 3% (1.1%)
  - Removed major code smells (A rating)



Overall	
Coverage	81.8%
Lines to Cover	1,442
Uncovered Lines	200
Line Coverage	86.1%
Conditions to Cover	468
Uncovered Conditions	147
Condition Coverage	68.6%
Tests	
Unit Tests	135
Errors	0
Failures	0
Skipped	0
Success	100%
Duration	2min

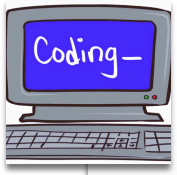
OUTLINE

- System functionality
- Job Quality
- **Process and teamwork**
- Technology transfer

PROCESS - AGILE AND SCRUM

Sprint 1

- Understand the team members
- Understand the product backlog and prepare UML, ERD and UI sketches
- Understand the Prattle code
- Achieve 85% coverage on Prattle legacy code



Sprint 2

- DB implementation
- Test message persistence
- sending messages to another user/group
- Provide online users
- Build Authentication system
- Develop basic user interface in react

Sprint 3

- Develop robust scalable architecture
- Develop GET and POST API endpoints
- Use design patterns: Factory method, Singleton, dependency injection
- Enable push notification
- Define architecture for react application
- Deploy on EC2

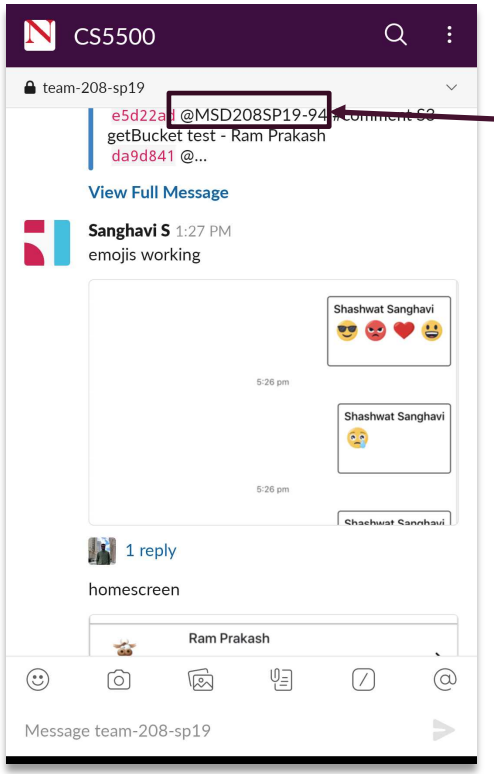
Sprint 4

- Integration
- Network communication library for frontend
- Resolve synchronization issues
- Add multimedia, emoji support
- Add language translation support
- Provide interface for all prattle features on the frontend application

We used JIRA and smart commits to keep record in each sprint.

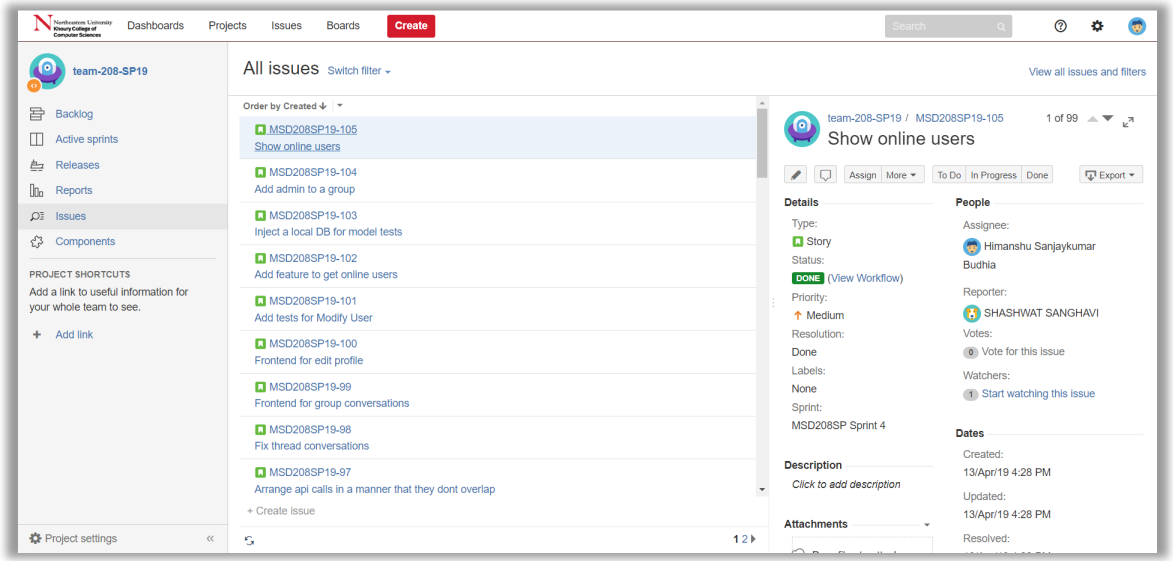
PROCESS - COMMUNICATION

- Meetings (Offline & telephonic)
  - Standup (every other day)
  - Telephonic (every day)
  - Every Wednesday, Saturday to solve each others road blocks
- Slack / WhatsApp



Smart commits

PROCESS – TOOLS

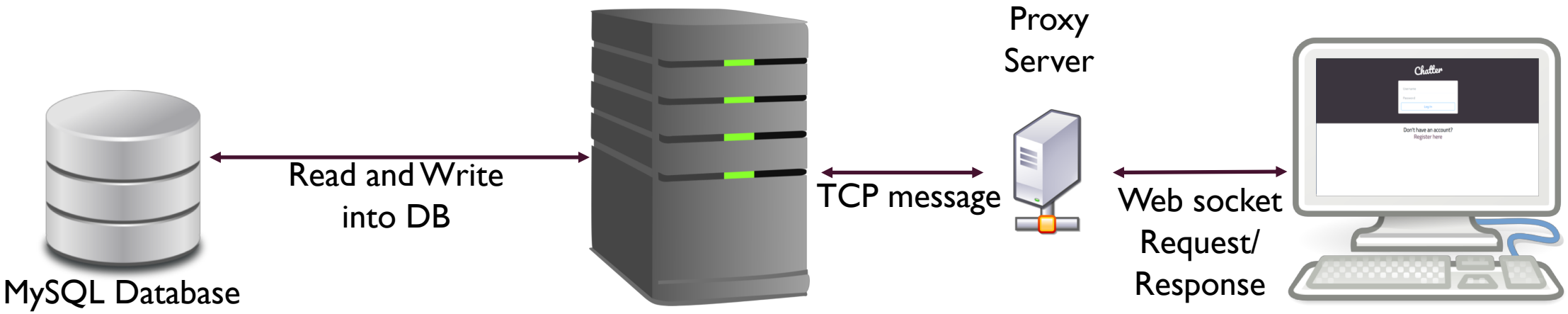


- JIRA – keep track of backlog
  - Helped to monitor individuals progress and roadblocks
- Continuous integration
  - Jenkins, Sonarqube
- Deploy environment
  - AWS

OUTLINE

- System functionality
- Job Quality
- Process and teamwork
- **Technology transfer**

TECH TRANSFER – BASIC ARCHITECTURE



TECH TRANSFER – BASIC ARCHITECTURE

- Backend uses MVC architecture
- Each message except signin (HLO) are sent with type API
- API message has route, method and data (JSON) in it
  - Eg. API <len(username)> <username> <len(msg)> <route>::<method>::<data>
    - API 3 xyz sendMessage/::POST::{username:'xyz', userid:'12', text:'Hello World!', mediaurl:null, receiver:'abc'}
- Design patterns
  - Factory method
  - Singleton
  - Observer pattern
  - Dependency Injection
- Testing
  - Junit
  - Mockito

TECHNOLOGY TRANSFER - DEPLOY

- Any AWS ec2 server can be used to host backend
- MySQL Database hosted on AWS RDS
  - Schema is delivered in the repository
- Amazon S3 is used to store static file content
- FRONTEND is a single page static application
  - possible to host on any web file server (Github can directly host it)
  - Uses flux architecture (widely used in the industry)

FUTURE WORK

- Searching in conversations
- Interface for government / super user
- Parental control
- Categorization of messages based on keywords or hashtags
- Forwarding messages
- Support for paid plans
- Server scaling for multiple users

THANK YOU