

Final Presentation Chatter



Sangeetha Chandrashekar, Jerry Lanning, Virat Goradia
Sibendu Dey, Tarun Kumar Attayampatty Sekar

System Functionality

Core Features provided in the application:

- User to user conversation
 - A user will be able to search for a user and message to the user directly
- Group messaging
 - A user will be able to search for a group and will be able to message in that group if he/she is part of the group.
 - A user will be able to perform other activities on the group such as viewing the group details, creating sub-group(moderator), setting password to a group(moderator).
 - Every user also has an option to a create a group and add/remove users from it.

System Functionality(contd)

- Invite System
 - Every user will be able to add users to a group if the user is the moderator of the group.
 - If a user wants to join a group, then the user must send a request invite to the moderator of the group who will accept or reject the invite.
- Profile
 - Each user has a seperate profile which can be updated by the user at any time.
 - The user can update their email address, image url, password.
 - The user also has an option to update his/her profile access to be set either public or private

System Functionality(contd)

- Circles
 - A user will be able to search for a user and will be able to follow or unfollow a user.
 - Once a user follows another user he/ she will be able to view the details of the user following.
 - If a user has set their profile is private, then the list of users following the user will not be able to access his/her details.

Programming Features Provided

- Hosted the app in the cloud.
- Created a client side part to the application as a jar which the user can run and access the various features of the app.
- Created custom exception for each kind of error thrown from the network layer of the app.
- Implemented the singleton pattern in all the JPA service classes and made use of the singleton in the Junit test cases.
- Made use of various design patterns – factory method, abstract pattern, etc which improved the code readability.

Security Features Provided

- The user email address and password are encrypted and saved in the database. Thus even if the DB is compromised, the data can not be decrypted unless the decryption key is known to the hacker. Java crypto cipher is used by our system for encryption.
- In order to provide security to a chat group, only the moderator of the group can accept or reject invitations.
- A user will be able to delete a message from his chat conversation and that message will be removed from that particular window view, but the deleted message will still be persisted in the database.
- A user will not be able to inject SQL query in any form in the entire application.

Is it useful for the client?

- Client is unable to enter invalid/incorrect inputs.
- Client is provided an appropriate error message whenever they enter any invalid/incorrect data or when they try to perform a functionality they do not have access to.
- The error message is in simple english making it easy for the client to understand on the high level as to what went wrong. They need not worry about the internal working of the system.
- Clients can easily navigate the system using numbers that map to a particular action they wish to perform. Example: The console displays: “1 - Chat with User.” Client just needs to enter “1” to be able to chat with some user.

Invalid Email

```
1 - Login
2 - Sign Up
* - QUIT
? - Help
?? - Download the help document
```

2

Enter Email Address:

virat@gmail.com

Enter Username:

Virat12~

Enter Password:

Virat12~

Confirm Password:

Virat12~

Enter new image URL:

<https://virat.com>

The email id you entered is invalid. Please try again! (Eg. youremailaddress@xyz.com)

Sign Up Failed

```
1 - Retry
0 - Go Back
* - Exit
|
```


Invalid image url

Enter Email Address:

virat.g@gmail.com

Enter Username:

Virat123

Enter Password:

Virat123

Confirm Password:

Virat123

Enter new image URL:

https://virat123.com

The imageURL you entered is invalid. Please try again! (Eg. http://* or https://*)

Sign Up Failed

1 - Retry

0 - Go Back

* - Exit

.

Invalid Username

Enter Email Address:

abcd@gmail.com

Enter Username:

Abc

Enter Password:

Abc

Confirm Password:

Abc

Enter new image URL:

<https://abc.com>

Username must be between 4 and 20 digits long, contain at least one number, one uppercase letter and one lowercase letter

Sign Up Failed

1 - Retry

0 - Go Back

* - Exit

.

Invalid password

Enter Email Address:

`abcd@gmail.com`

Enter Username:

`Abcd123`

Enter Password:

`Abc`

Confirm Password:

`Abc`

Enter new image URL:

`https://abcd.com`

Password must be between 4 and 20 digits long, contain at least one number, one uppercase letter and one lowercase letter.

Sign Up Failed

1 - Retry

0 - Go Back

* - Exit

|

Incorrect Password

Enter Email Address:

abcd@gmail.com

Enter Username:

Abcd123

Enter Password:

Abcd123

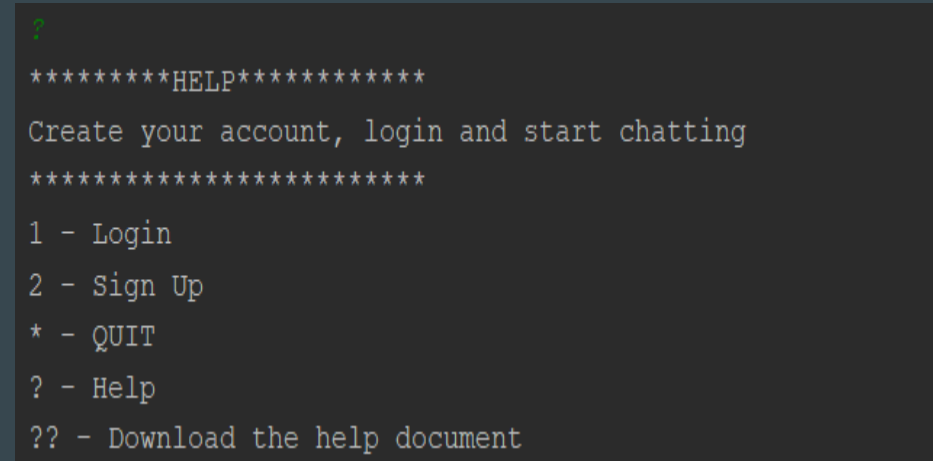
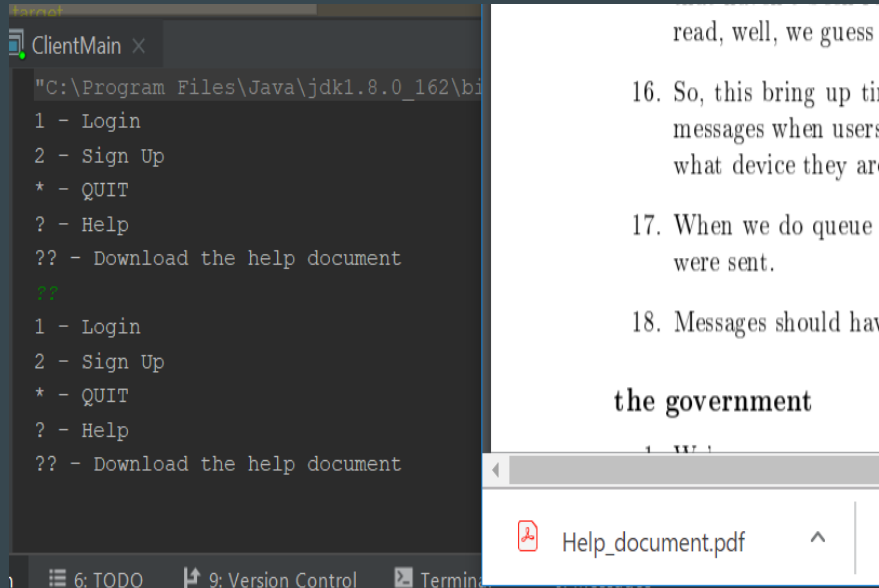
Confirm Password:

Abcd122

Passwords do not match. Renter password

Is it useful for the client?

- Client is provided with a help window which helps them understand how to interact with the system and what different functionalities are available to them.



Is it useful for the client?

- Client is able to time shift, that is, when they come back online, it looks like the service is picking up from where they left off.
- Client can interact with the system with just the basic understanding of simple English language.
- Client can make their profiles either public or private. This enables the client to either be found or not. Therefore, they get control in their hands, if they want to keep themselves visible to others or not, giving them a good feel of security.
- Client gets a separate window for their respective personal or group conversations.

Message windows

```
Enter user name:  
Virat123  
1)Virat123:Hi Abcd  
  
Hi Virat123  
2)you:Hi Virat123  
|
```

```
Enter user name:  
Abcd123  
  
Hi Abcd  
1)you:Hi Abcd  
2)Abcd123:Hi Virat123  
|
```

Is there evidence from the backlog/requirements to support these claims?

Backlog

Board ▾

⌵

QUICK FILTERS: Only My Issues Recently Updated

VERSIONS

EPICS

▼ MSD108SP Sprint 4 8 of 24 issues visible [Clear all filters](#) ACTIVE

0

0





























26

⋮





























... 01/Apr/19 9:23 PM • 22/Apr/19 9:23 PM

	↑	MSD108SP19-169	Refactor MessageJPAService/Impacted Layers/Tests		5
	↑	MSD108SP19-171	Refactor Profile JPAService/Impacted Layers/Tests		5
	↑	MSD108SP19-178	Add javadocs to all service classes		2
	↑	MSD108SP19-179	Refactor User related files to use common AllJPAService methods		5
	↑	MSD108SP19-180	Refactor Group related files to use common AllJPAService methods		5
	↑	MSD108SP19-181	remove minor code smells		1
	↑	MSD108SP19-182	Refactor addUser method		1
	↑	MSD108SP19-185	create AllJPAService class		2



























Is there evidence from the backlog/requirements to support these claims?

MSD108SP19-55 *	Exception Handling	 Story	 Medium	DONE	- → 4
MSD108SP19-110 *	Set up invite class	 Story	 Medium	DONE	- → 2
MSD108SP19-111 *	Create JPA for invite class	 Story	 Medium	DONE	- → 2
MSD108SP19-113 *	Create custom exceptions for Profile	 Story	 High	DONE	- → 1
MSD108SP19-114 *	Create custom exceptions for User	 Story	 High	DONE	- → 1
MSD108SP19-115 *	Create custom exceptions for Message	 Story	 High	DONE	- → 1
MSD108SP19-116 *	Create custom exceptions for Group	 Story	 High	DONE	- → 1
MSD108SP19-117 *	Add custom exception handling in groupJPAService file	 Story	 High	DONE	- → 3
MSD108SP19-118 *	Add custom exception handling in profileJPAService file	 Story	 High	DONE	- → 2
MSD108SP19-119 *	Add custom exception handling in messageJPAService file	 Story	 High	DONE	- → 3
MSD108SP19-120 *	Add custom exception handling in userJPAService file	 Story	 High	DONE	- → 3
MSD108SP19-121 *	Add custom exception handling in groupService file	 Story	 Medium	DONE	- → 2
MSD108SP19-122 *	Add custom exception handling in messageService file	 Story	 Medium	DONE	- → 2
MSD108SP19-123 *	Add custom exception handling in profileService file	 Story	 Medium	DONE	- → 1



























Is there evidence from the backlog/requirements to support these claims?

MSD108SP19-124 *	Add custom exception handling in userService file	 Story	 Medium	DONE	- → 2
MSD108SP19-125 *	Add custom exception handling in groupController file	 Story	 Medium	DONE	- → 1
MSD108SP19-126 *	Add custom exception handling in userController file	 Story	 Medium	DONE	- → 1
MSD108SP19-127 *	Add custom exception handling in profileController file	 Story	 Medium	DONE	- → 1
MSD108SP19-128 *	JUNIT tests for profile controller	 Story	 Medium	DONE	- → 1
MSD108SP19-129 *	JUNIT tests for group controller	 Story	 Medium	DONE	- → 1
MSD108SP19-130 *	JUNIT tests for profile service	 Story	 Medium	DONE	- → 2
MSD108SP19-131 *	JUNIT tests for group service	 Story	 Medium	DONE	- → 2
MSD108SP19-132 *	JUNIT tests for message service	 Story	 Medium	DONE	- → 2
MSD108SP19-133 *	JUNIT tests for message JPA service	 Story	 Medium	DONE	- → 3
MSD108SP19-134 *	JUNIT tests for profile JPA service	 Story	 Medium	DONE	- → 3
MSD108SP19-135 *	JUNIT tests for group JPA service	 Story	 Medium	DONE	- → 3
MSD108SP19-136 *	Edit the createlfNotPresent method of groupService	 Story	 High	DONE	- → 3
MSD108SP19-137 *	JUNIT tests for createlfNotPresent method of	 Story	 Medium	DONE	- → 2

Is there evidence from the backlog/requirements to support these claims?

MSD108SP19-138 *	Update password for user	 Story	 Medium	DONE	- → 2
MSD108SP19-139 *	User must be able to unfollow users	 Story	 Medium	DONE	- → 3
MSD108SP19-140 *	User must be able to create sub groups	 Story	 Medium	DONE	- → 3
MSD108SP19-141 *	User must be able to view and edit the detail of the sub group	 Story	 Medium	DONE	- → 3
MSD108SP19-142 *	Create an invalid email exception	 Story	 Highest	DONE	- → 1
MSD108SP19-143 *	JUNIT for invalid email exception	 Story	 High	DONE	- → 1
MSD108SP19-145 *	Handling Exceptions for Invitation	 Story	 Medium	DONE	- → 4
MSD108SP19-146 *	Create controller and service methods for invite system	 Story	 Medium	DONE	- → 4
MSD108SP19-147 *	Test all user exceptions	 Story	 Medium	DONE	- → 8
MSD108SP19-148 *	Create the failed login messages	 Story	 Medium	DONE	- → 4
MSD108SP19-149 *	Mock the database for invitation JPA Service	 Story	 Medium	DONE	- → 2
MSD108SP19-150 *	Invitation System (client)	 Story	 Medium	DONE	- → 5
MSD108SP19-151 *	Private conversation	 Story	 Medium	DONE	- → 8

Is there evidence from the backlog/requirements to support these claims?

MSD108SP19-150 *	Invitation System (client)	 Story	 Medium	DONE	- → 5
MSD108SP19-151 *	Private conversation	 Story	 Medium	DONE	- → 8
MSD108SP19-152 *	RequestDispatcher Tests for invitation system	 Story	 Medium	DONE	- → 2
MSD108SP19-153 *	RequestDispatcher code and junit refactoring	 Story	 Medium	DONE	- → 8
MSD108SP19-154 *	JsonBufferedReader refactor	 Story	 Medium	DONE	- → 2
MSD108SP19-156 *	Add delete group feature	 Story	 Medium	DONE	- → 3
MSD108SP19-157 *	Fix bugs related to User and moderators of group	 Story	 Medium	DONE	- → 4
MSD108SP19-158 *	Integration testing with group delete feature	 Story	 Medium	DONE	- → 2
MSD108SP19-159 *	Fix bugs related to invite group	 Story	 Medium	DONE	- → 2
MSD108SP19-162 *	Basic integration testing	 Story	 Medium	DONE	- → 15
MSD108SP19-163 *	Create Invalid ImageURL Exception	 Story	 Medium	DONE	- → 1
MSD108SP19-164 *	JUNIT for invalid image URL	 Story	 Medium	DONE	- → 1
MSD108SP19-165 *	Client side Invalid image url exception handling	 Story	 Medium	DONE	- → 1

Job Quality

- Designed the entire code base using MVC architecture to have clear bifurcation between the implementation layers.
- Added JavaDocs for all classes and methods to have a better understanding of the code functionality.
- Added SonarLint to development environment to avoid code smells while coding.
- Every new feature was tested using Junit test cases and integration tests.
- Every new feature had a separate branch on GitHub. To merge to other branches, we used git pull requests and facilitate peer acknowledgments.

Job Quality

- A pseudo-master branch was implemented every sprint and all the code merges to this branch were done using PR. This helped us to maintain constant peer review for every code merge and nip the problem in the bud, if any.
- These pseudo-master branches also had CI tool Jenkins enabled to it to track the code quality.
- Feedback from the sprint review report helped us further improve the code quality.

Job Quality

- Constantly monitored Jenkins and SonarQube report to assure 85% or above code coverage, fixing bugs and eliminating code smells.

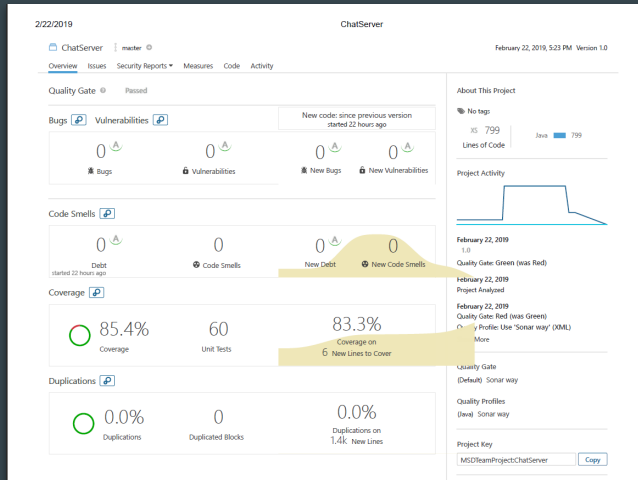


Fig 1. Sprint 1 SonarQube report (60 tests)

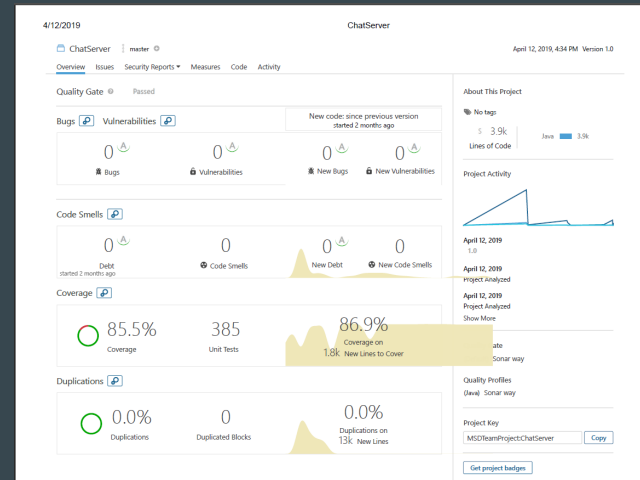


Fig 2. Sprint 4 SonarQube report (385 tests)

Job Quality

- Had a dedicated sprint to refactor the code to have better readability, wherever it was deemed necessary.
- Refactored the code to include design patterns such as singleton and strategy pattern etc.

[illegible]

Sprint 2 RequestDispatcher.java

Code snippet

[illegible]

Sprint 4 RequestDispatcher.java code snippet with strategy pattern

Process and Teamwork

- Our process involved splitting the team into a couple teams.
 - One working on the client side.
 - The other team working on the controller and service layers.
- Development Branch on GitHub
 - Everyone creates PRs to a single branch and everyone contributes to the same code base to avoid merge conflicts.
- Separation of Tasks
 - Each team had their own tasks in the process and were able to focus on ensuring their tasks were working.
- Focus on the Goals with more dependencies.
 - We don't want other people affected because we haven't completed a specific task.
- Managing Our Course Load
 - Ensuring people with heavier weeks were given lighter loads when they needed it.

Working as a team

- Each member was open about their workload.
 - People with heavier workloads wouldn't have to as many tasks as someone with a lighter workload that week. This came with the expectation they would do more work in the week they had a lighter workload.
- Each member was honest about what tasks they were comfortable with.
 - Assigning tasks to people with the most experience in each task.
- Daily Scrums
 - Working in daily 10-15 min scrums helped us communicate progress
- Code Meetings
 - Sitting down and coding together allowed for easy communication.

The Process, Did it Work?

- The process as a whole worked.
- Completing tasks with more dependencies.
 - Completing tasks with more dependencies early led to less frustration between team members.
- Ensure tested code is pushed to master.
 - This lead to code that didn't break when other people used it.
- Daily Scrums
 - Communication is KEY, the daily scrums kept us on track.

Automation

- Work in a build, test and promote process.
 - We started by building a feature.
 - After building a feature we put it through the testing phase.
 - If the build failed tests we needed to alter the way we built the code and refactor the existing code.
 - After passing our tests we could push our code to a develop branch for others to try to find bugs
 - If no bugs were found we thought about different ways to refactor the code to be more efficient.
 - After everyone tried to break our code and gave their input we could push the develop branch to master.

Recognize Short Comings

- Separation of Tasks
 - We couldn't all just work on the list of Goals, we had to separate the list of goals into individual tasks.
- Separation of layers
 - Allowed us to more efficiently develop code.
- Too many Goals/Commitments!
 - After a couple sleepless nights we learned how to better predict our work load.
- Merge Conflicts!
 - Many merged conflicts arose, by creating a develop branch everyone branched off of it allowed us to create PR's and recognize merge conflicts before they happened!
- Ask for Help!
 - If you're feeling overwhelmed the best thing to do is ask for help!

Technology Transfer

- Extensible codebase.
 - Good design principles makes way for a good future development.
- Live system up and running.
 - Live system has an embedded database for testing the product functionalities.
- System demo video.
 - Shows overview of the core functionality.
- System setup video.
 - Tutorial to demonstrate the steps for running the system.

Future Enhancements to the Application

- Reply to particular message in separate thread.
- Forwarding message to other groups.
- Tracking the source of the message that has been forwarded.
- Sending and receiving attachments.
- Sending auto-delete messages to a user/group.
- Providing localisation in the app that supports various languages.

Any Questions???

Please contact us at the following email addresses:

Jerry Lanning <lanning.j@husky.neu.edu>

Sangeetha Chandrashekar <chandrashekar.s@husky.neu.edu>,

Sibendu Dey <dey.s@husky.neu.edu>,

Virat Goradia <goradia.v@husky.neu.edu>,

Tarun Kumar Attayampatty Sekar <attayampattysekar.t@husky.neu.edu>