

50.003 Elements of Software Construction Meeting

Report 2

Cohort 3 Group 4

Yap Wei Lok, Daniel Chin, Goh Yujin, Joel Chua

20 Feb 2018

Changes in requirements

Primary requirements Our primary requirements like real time chat and concurrency still remain. Multiple users would have to be able to join public groups anytime or private groups via invites, be able to send messages to the chat rooms in said groups and also receive messages at the same time.

Secondary requirements Requirements like group-finding via tags and one-off match-making remain the same as well. Users should be able to find groups that they are interested in via tags. For example, someone interested in learning more about foreign cultures can search for *#japanese*, *#korean*, *#italian* etc. Furthermore, when users want to look for one-off groups to play some sports or games, the user can use the matchmaking system to find people that are interested in doing so.

Development platform After some testing and researching, we have decided on using *Meteor.js* to develop our web application project. Meteor.js is a full stack Javascript framework that abstracts away some of the boiler plate code for both front-end and back-end, enabling us to focus on the business logic.

Formal documentation of uses cases

We have laid out a few detailed use case documentation in the appendix, namely Announcement System (Table 1), Polling System (Table 2), Chat Group (Table 3) and Interactive Classroom (Table 4). We have also appended a combined use-misuse case diagram for our project below.

insert use-misuse case diagram here

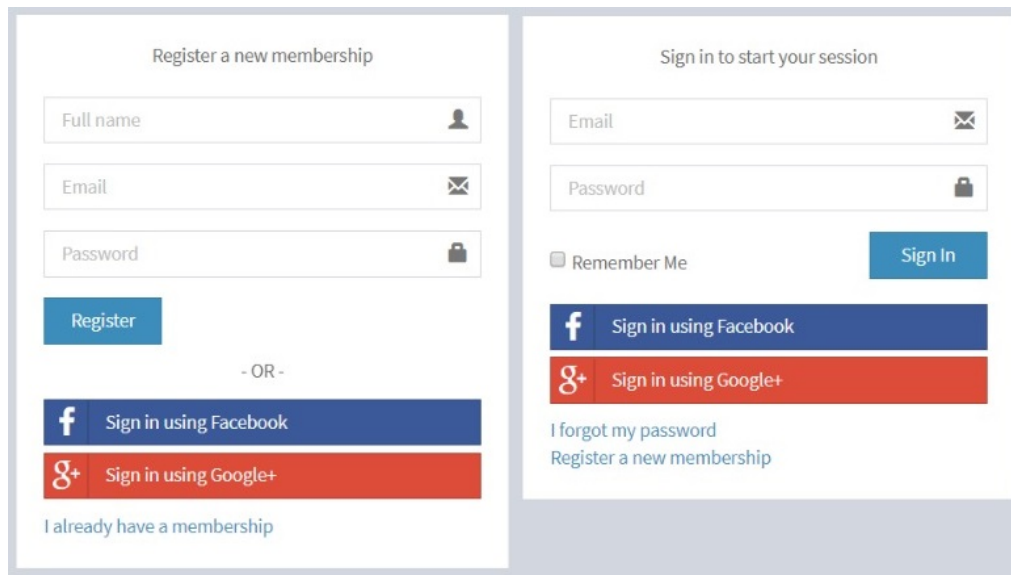
Initial design

insert class diagram here

Implementation of basic features

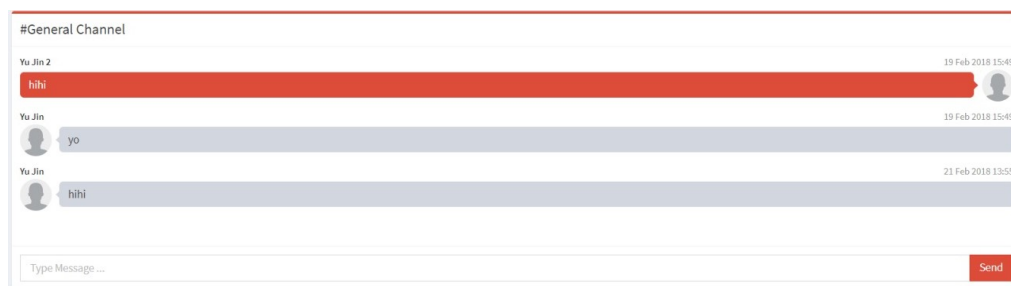
Using Meteor, we rapidly prototyped a single chat room with login and tested it with two users. The users are able to register a new account and login to an existing account at the login page as shown in Figure 1 below. In the future, we hope to incorporate SUTD's login system so the users would not have to register and can just use their SUTD account for our web application.

We registered two users on different browsers and sent some messages to the same chatroom as seen in Figure 2 below. The two users were able to send and receive the messages, so the the chatroom prototype can be considered a success.



The image displays two side-by-side web forms. The left form, titled 'Register a new membership', includes input fields for 'Full name', 'Email', and 'Password', each with a corresponding icon (person, envelope, and lock). Below these fields is a blue 'Register' button. Underneath the button is a separator '- OR -' and two social login options: 'Sign in using Facebook' (with a Facebook 'f' icon) and 'Sign in using Google+' (with a Google+ 'g+' icon). At the bottom of the left form is a link 'I already have a membership'. The right form, titled 'Sign in to start your session', includes input fields for 'Email' and 'Password' with envelope and lock icons. Below these is a 'Remember Me' checkbox and a blue 'Sign In' button. Further down are two social login buttons: 'Sign in using Facebook' (blue with 'f' icon) and 'Sign in using Google+' (red with 'g+' icon). At the bottom of the right form are two links: 'I forgot my password' and 'Register a new membership'.

Figure 1: Registration (Left) and Login (Right)



The image shows a chatroom interface titled '#General Channel'. It displays a list of messages from two users, 'Yu Jin 2' and 'Yu Jin'. The first message is from 'Yu Jin 2' with the text 'hihi' and a timestamp of '19 Feb 2018 15:49'. The second message is from 'Yu Jin' with the text 'yo' and a timestamp of '19 Feb 2018 15:49'. The third message is from 'Yu Jin' with the text 'hihi' and a timestamp of '21 Feb 2018 13:55'. At the bottom of the chatroom is a text input field labeled 'Type Message ...' and a red 'Send' button.

Figure 2: Chatroom with two concurrent users

Testing plan

Database We plan to test the database API query functions by creating some unit tests to push and pull information from the database.

Authentication The correct user and password combination should return the correct data from the database. In this case, we can test the authentication of the login system and also whether we get the correct information like joined groups and messages.

Search The search input has to be parsed properly and the search engine has to be able to retrieve the relevant items. For example *foo* input should return groups with tags like *#football* or *#food*.

Announcement System Given that most announcements are given an expiry time, we have to test that the announcements do disappear in the client-side. Furthermore, non-privileged users in a group should not be able to make announcements in said group.

Polling System Similar to the announcements, polls also have a time or vote limit. We would also have to test that the polls are sent to the correct group and the poll questions are as entered by the creator. The poll results also have to be tested to ensure that all poll entries are correctly stored in the database.

Spam Test We plan to create a script to spam the chat with a bunch of messages. Our current idea to counter this is to block messages from being sent if too many are sent in a given time frame.

Appendix

Name	Announcement System
Objective	Disseminate information to user that are subscribed to a group/channel
Pre-conditions	<ol style="list-style-type: none"> 1. Users must be subscribed to a group/channel 2. Announcements can only be made by privileged users
Post-conditions	<p>Success</p> <ol style="list-style-type: none"> 1. Announcement sent is successfully received by subscribed users <p>Failure</p> <ol style="list-style-type: none"> 1. Announcement is not sent to the server 2. The subscribed users do not receive the announcement
Actors	<p>Primary</p> <ol style="list-style-type: none"> 1. Announcer <p>Secondary</p> <ol style="list-style-type: none"> 1. Server 2. Subscribed users
Trigger	Announcement event created by privileged user
Normal Flow	<ol style="list-style-type: none"> 1. Privileged user creates an announcement via the CreateAnnouncement function 2. Poll is sent to the server upon success of CreateAnnouncement 3. Handshake between announcer and server upon successful dispatch of announcement message 4. The server stores the message in the database 5. When a subscribed user is online, the server fetches the message from the database 6. The server serves the message to the user 7. Handshake between server and client upon successful receipt of message 8. Remove the message from the database after X weeks
Alternative Flow	8a. Remove the message from the database after Y time if timeout duration is specified
Interacts With	CreateAnnouncement, Database
Open Issues	<ol style="list-style-type: none"> 1. How long should the message be stored in database if no timeout is specified? 2. How should the system prevent announcer from spamming messages?

Table 1: Announcement System use case table

Name	Polling System
Objective	Poll users that are subscribed to a group/channel
Pre-conditions	<ol style="list-style-type: none"> 1. Users must be subscribed to a group/channel 2. Poll can only be created by privileged users
Post-conditions	<p>Success</p> <ol style="list-style-type: none"> 1. Users are able to view and respond to the poll <p>Failure</p> <ol style="list-style-type: none"> 1. Poll content not sent to the server 2. Subscribed users unable to view or interact with the poll/survey
Actors	<p>Primary</p> <ol style="list-style-type: none"> 1. Poll creator <p>Secondary</p> <ol style="list-style-type: none"> 1. Server 2. Subscribed users
Trigger	Poll event created by privileged users
Normal Flow	<ol style="list-style-type: none"> 1. Privileged user creates an announcement via the CreatePoll function 2. Poll is sent to the server upon success of CreatePoll 3. Handshake between poll creator and server upon successful dispatch of poll 4. The server stores the poll in the database 5. When a subscribed user is online, the server fetches the poll from the database 6. The server serves the poll to the user 7. Handshake between server and client upon successful receipt of poll 8. User responds to the poll; User votes are sent to the server after poll submission 9. Handshake between user and server upon successful receipt of poll results 10. Remove the poll from the database after X weeks
Alternative Flow	<ol style="list-style-type: none"> 8a. User submits the poll without completing it; Prompt user to complete 10a. Remove the poll after Y time if timeout duration is specified 10b. Remove the poll after Z amount of votes if max vote number is specified
Interacts With	CreatePoll, Database
Open Issues	<ol style="list-style-type: none"> 1. How long should the message be stored in the database if no timeout is specified? 2. How do the poll creators access the results?

Table 2: Polling System use case table

Name	Chat Group
Objective	A private or public group for like-minded individuals to interact with each other
Pre-conditions	1. Chat group can only be created by a system administrator 2. A user has to request for a new chat group via RequestGroup function
Post-conditions	<p>Success</p> <p>1. Users are able to use the chat group for whatever intended purposes</p> <p>Failure</p> <p>1. Users are unable to access the chat group</p> <p>2. Users are unable to use some of the functions in the chat group</p>
Actors	<p>Primary</p> <p>1. A user or group of users</p> <p>2. System admin</p> <p>Secondary</p> <p>1. Server</p> <p>2. Other users</p>
Trigger	Chat group requested by a user
Normal Flow	<p>1. System admin creates a new chat group via CreateGroup function</p> <p>2. System admin appoints the user or group of users as group administrators</p> <p>3. A new Group object is created in the server</p> <p>4. Group admins set up the group and group tags</p> <p>5. Other users join the group via search</p> <p>6. Group members can send messages in the chatrooms</p> <p>7. When a group member are online, the server fetches new messages from the database</p> <p>8. The new messages are displayed on the group member's chat window</p> <p>9. Group admins can escalate or de-escalate privileges of the group members</p> <p>10. Group admins and privileged group members can create announcements and polls via CreateAnnouncement and CreatePoll functions respectively</p>
Alternative Flow	5a. If the group is private, other users can join via invite links
Interacts With	RequestGroup, CreateGroup, CreateAnnouncement, CreatePoll, Database

Table 3: Chat Group use case table

Name	Interactive Classroom
Objective	Private group/channel for students to ask questions or answer quizzes during class time
Pre-conditions	1. Interactive classroom can only be created by a system administrator 2. The instructor has to request for a Classroom via RequestClassroom function
Post-conditions	<p>Success</p> <p>1. Users are able to use the interactive classroom to facilitate their learning</p> <p>Failure</p> <p>1. Users are unable to access the interactive classroom</p> <p>2. Users are unable to use some of the functions in the interactive classroom</p>
Actors	<p>Primary</p> <p>1. Instructor</p> <p>2. System admin</p> <p>Secondary</p> <p>1. Server</p> <p>2. Students</p>
Trigger	Classroom requested by the instructor
Normal Flow	<p>1. System admin creates a classroom via the CreateClassroom function</p> <p>2. System admin appoints the instructor as the group administrator 3. A new special Group object is created in the server</p> <p>4. Instructor sends invitation links to his/her students</p> <p>5. Students receive the invitation and join the classroom</p> <p>6. The classroom function like any other chat group but with some additional features like quizzing and feedback forms</p>
Alternative Flow	<p>4a. Instructor displays the classroom code during class</p> <p>5a. Students join the classroom via the room code</p>
Interacts With	RequestClassroom, CreateClassroom, CreateAnnouncement, CreatePoll, CreateQuiz, CreateFeedback, Database

Table 4: Interactive Classroom use case table