Managing Software Development
Sprint 1 - Documentation

Team:

Number: 209

Members: Akhilesh Hegde, Amandeep Singh, Elavazhagan Sethuraman, Varsha Muroor Rao, Vinayakaram Nagasubramanian

Overview:

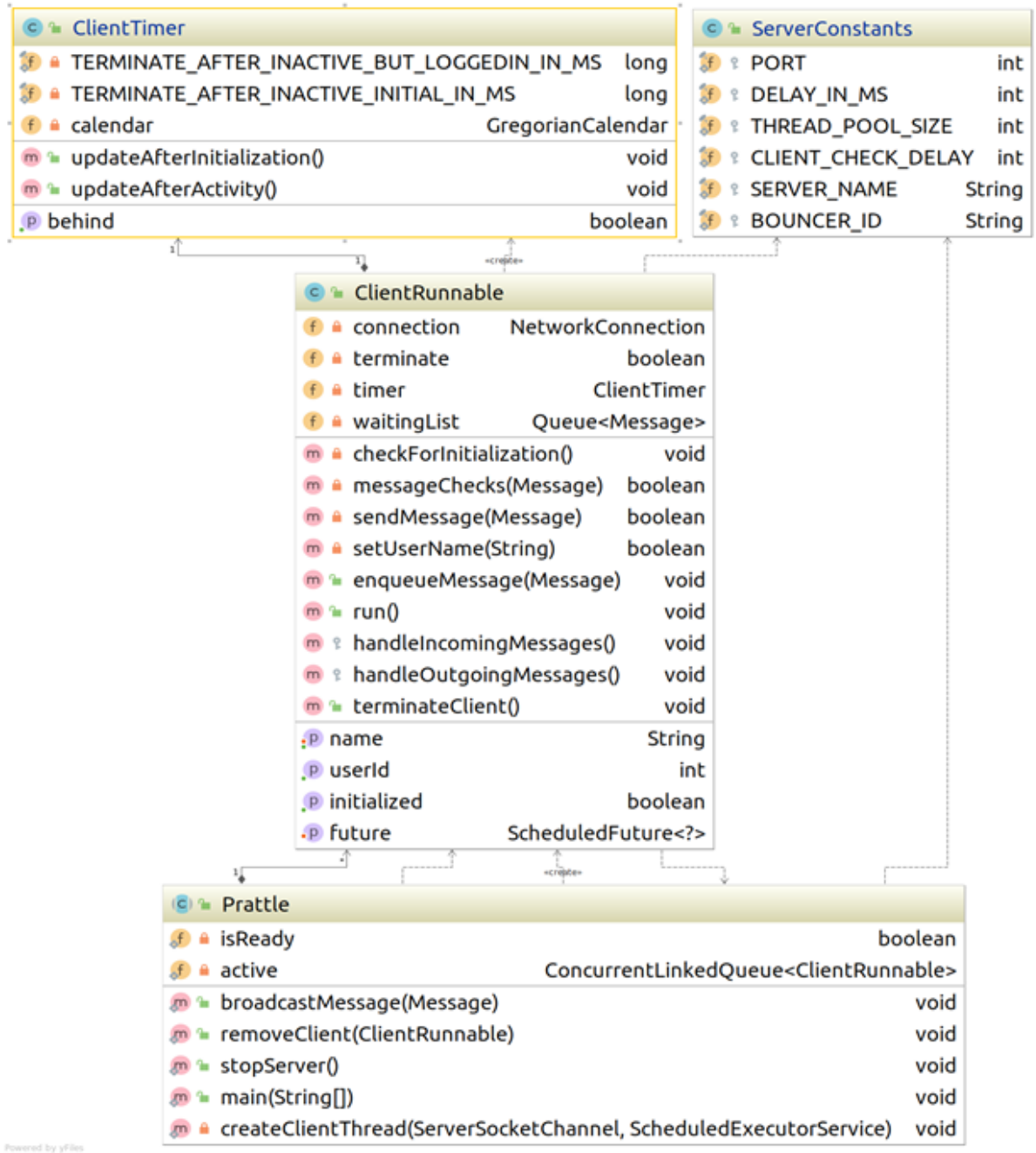
The first sprint lasted for a duration of 2 weeks starting from 2/11/2018 to 2/22/2018. Throughout the sprint, we had 3 planned group meetings and a few impromptu meetings and peer reviews. We achieved the following goals:

1. Generating the UML diagram for the given Prattle application, which clearly outlines the architecture of the server.
2. Created the UML class diagram for the application we're developing, outlining the functionality and entities involved in the product.
3. Created use case diagram which depicts the users of the system along with the actions that they can perform while using the system.
4. Set up Jira for sprint 1 with a sprint board and selecting a subset of the backlog items for include in the sprint scope.
5. Set up SlackBot for daily scrum meetings.
6. Created unit test cases for the given Prattle code, so as to achieve 92% statement coverage and the required condition coverage.
7. Merge the tested code to the master in GitHub.
8. Creating backlog items in Jira for the next sprint.

UML Class Diagram for the Prattle Server application:

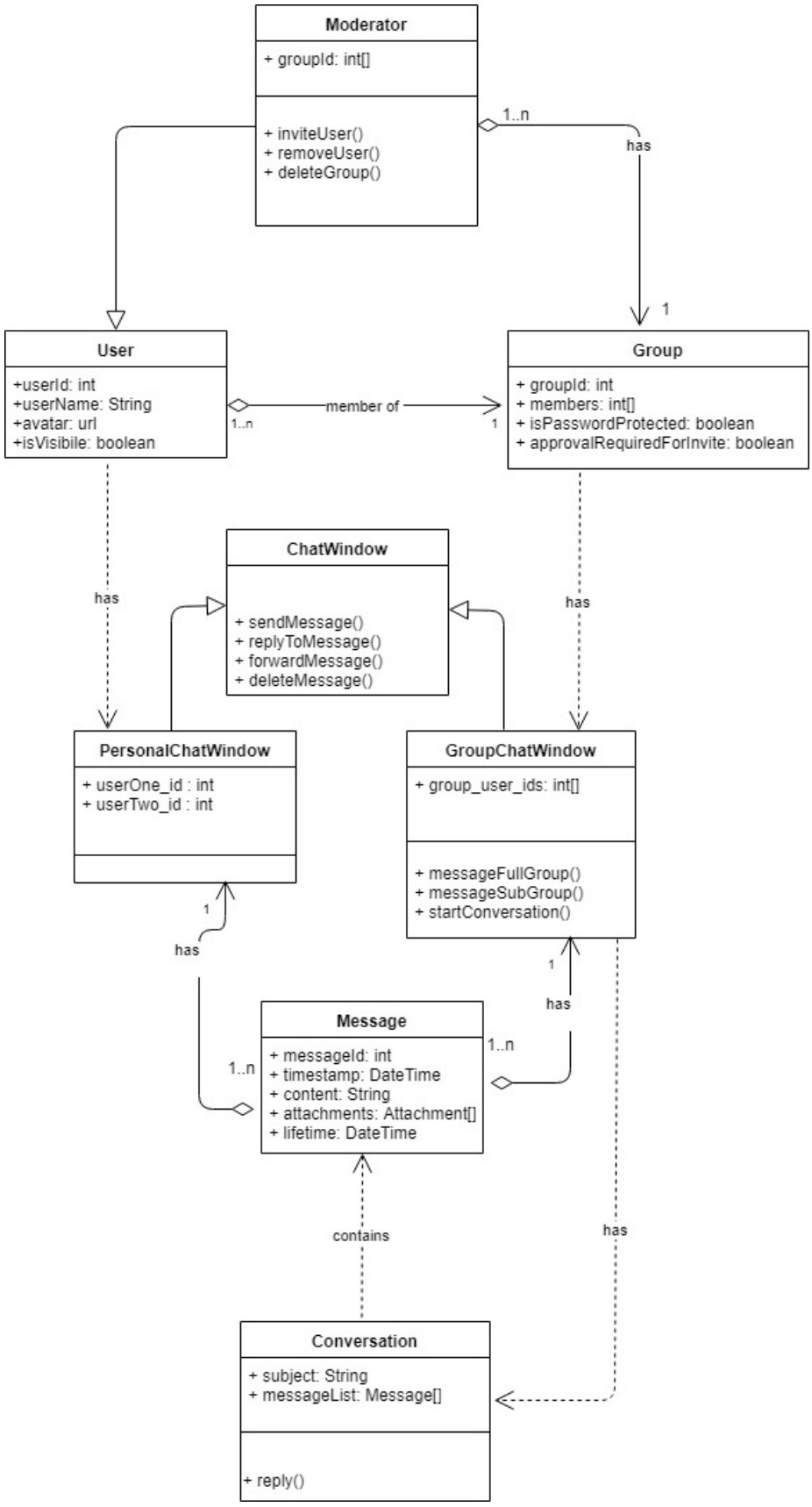
The UML class diagram for the Prattle application is generated using built-in tools in IntelliJ and Eclipse.

UML class diagram for the classes in server package (Prattle):



UML class diagram for the classes in IM package:

UML Class Diagram for the Client application (Chatter):



The application has the following entities:

1) **User**: The user of a system can have properties like userId, userName, avatar (icon) along with a boolean property isVisible which indicates whether this user is visible to be found by other groups and members. Users can also be a member of multiple groups.
2) **Moderator**: The moderator entity inherits from the User entity, and has additional properties that extend the functionality of a regular group user, which indicates the groups that the user moderates (is an admin).
3) **Chat-Window**: Chat-Window entity can have the following functions - send message, reply to message, forward message, delete message.
4) **Personal-Chat-Window**: A Personal-Chat-Window is inherited from the Chat-Window class and can have two properties; userOne_id - the user id of the user who has logged in, userTwo_id - the user id of the user with whom the logged-in user is chatting with.
5) **Group-Chat-Window**: A Group-Chat-Window is inherited from the Chat-Window class and has the property group_user_ids which is a list of the ids of the users who are a part of the group. The functions like messaging full group, messaging a subset of the group, and starting a conversation(thread) can be performed in a Group-Chat-Window.
6) **Message**: A Message entity has the properties like message id, timestamp, lifetime, content(text), and attachments.
7) **Group**: A group entity has the attributes like group id, members, a boolean - 'approvalRequiredForInvite' - indicating whether moderator permission is required to invite other users, and a boolean 'isPasswordProtected' which indicates whether an additional password is needed to open the group. A group can have 1 or more moderators and a single Group-Chat-Window.
8) **Conversation**: A conversation entity(message thread), belongs to the Group-Chat-Window and can have a subject, a list of messages, and a method reply.

Use Case Diagram for the application

The actors and use cases for each and a corresponding use case diagram are shown below.

1) **For a User**

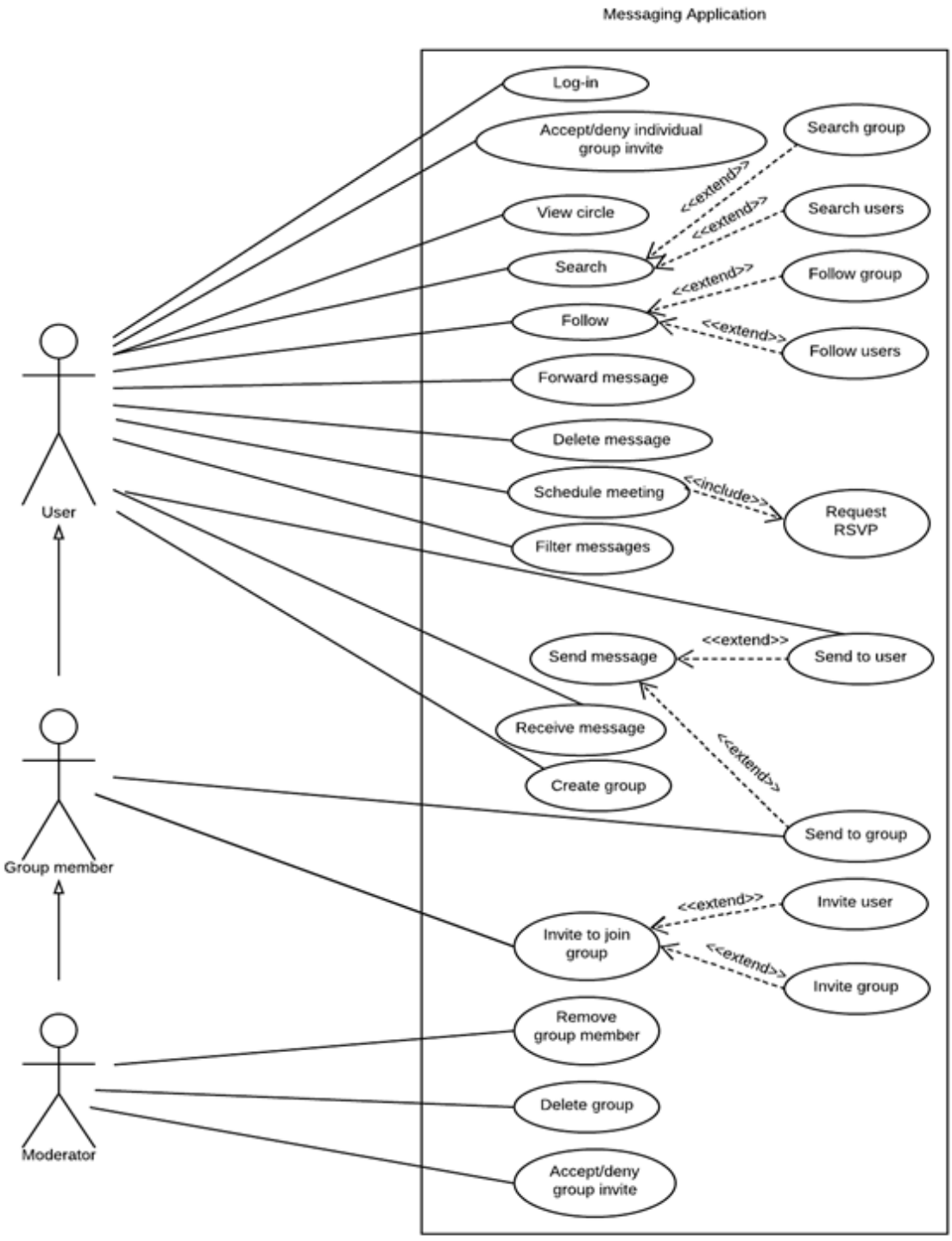
- Log in to the application
- Send message to another user
- Receive message
- View circle
- Create a new group
- Search other users/groups
- Follow other users/groups
- Forward message
- Delete message
- Accept/deny group join request
- Filter messages
- Schedule meetings and request for RSVP

2) **For a Group member**

- Send messages to group
- Invite user/other group to join the group

3) **For a Moderator** (extends group member functionality)

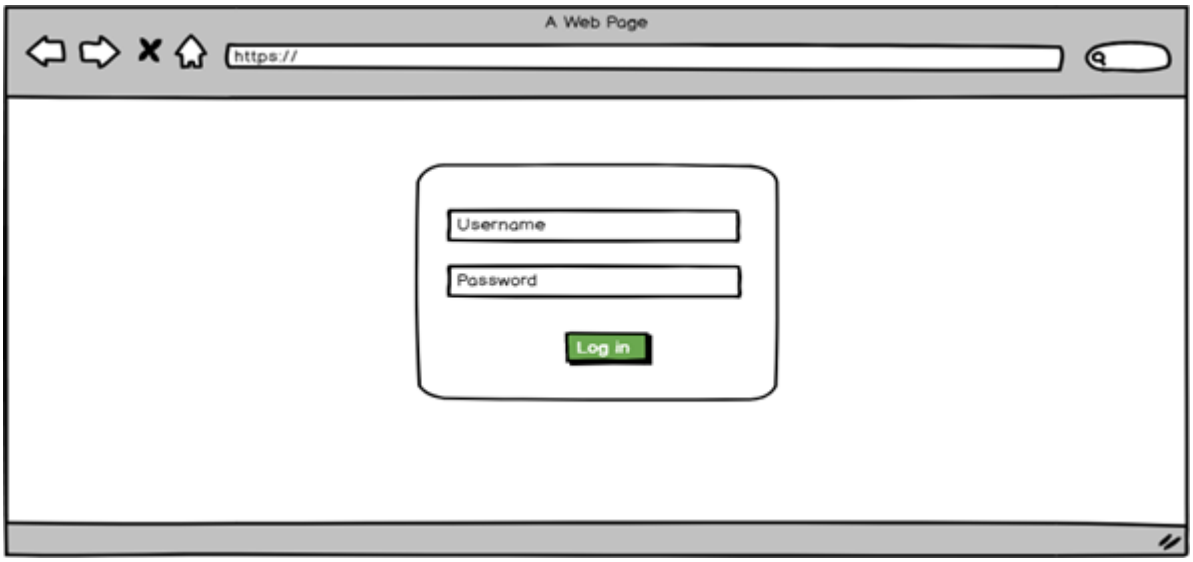
- Remove a group member
- Accept/deny group invitation
- Delete group


Messaging Application
Log-in
Accept/deny individual group invite
View circle
Search
Follow
Forward message
Delete message
Schedule meeting
Filter messages
Send message
Receive message
Create group
Invite to join group
Remove group member
Delete group
Accept/deny group invite
Search group
Search users
Follow group
Follow users
Request RSVP
Send to user
Send to group
Invite user
Invite group
<<extend>>
<<include>>
User
Group member
Moderator

Wireframes

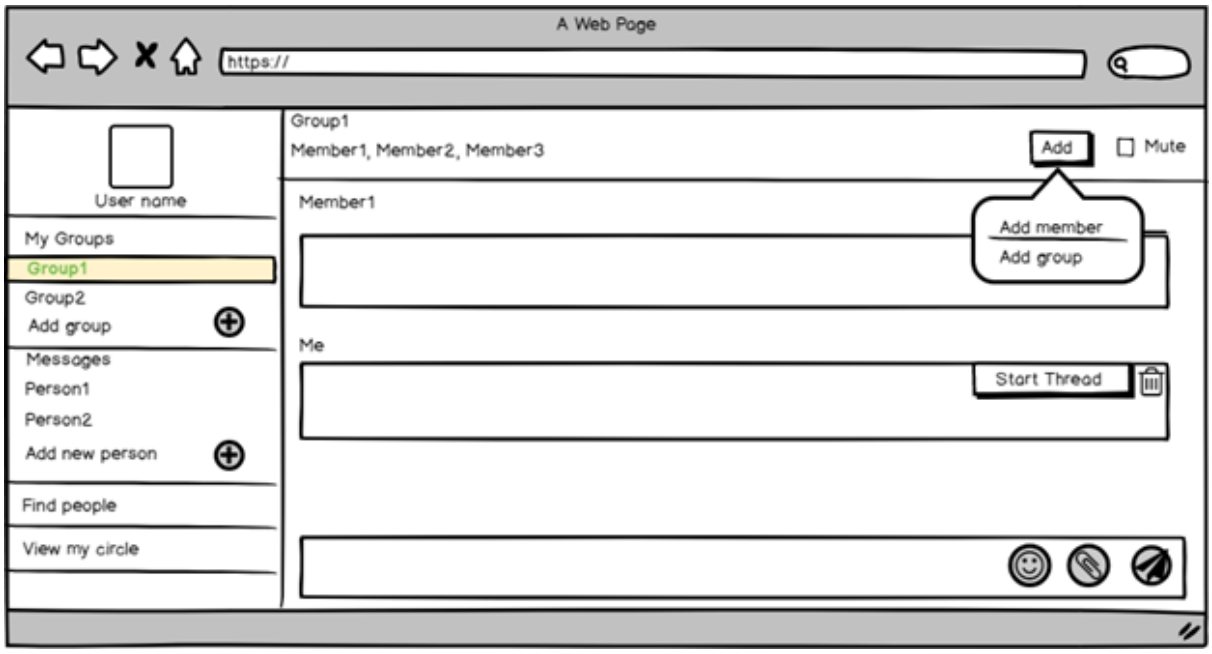
The below images give a rough idea about what the product can look like. These designs were created using the free online tool Balsamiq.

Login screen

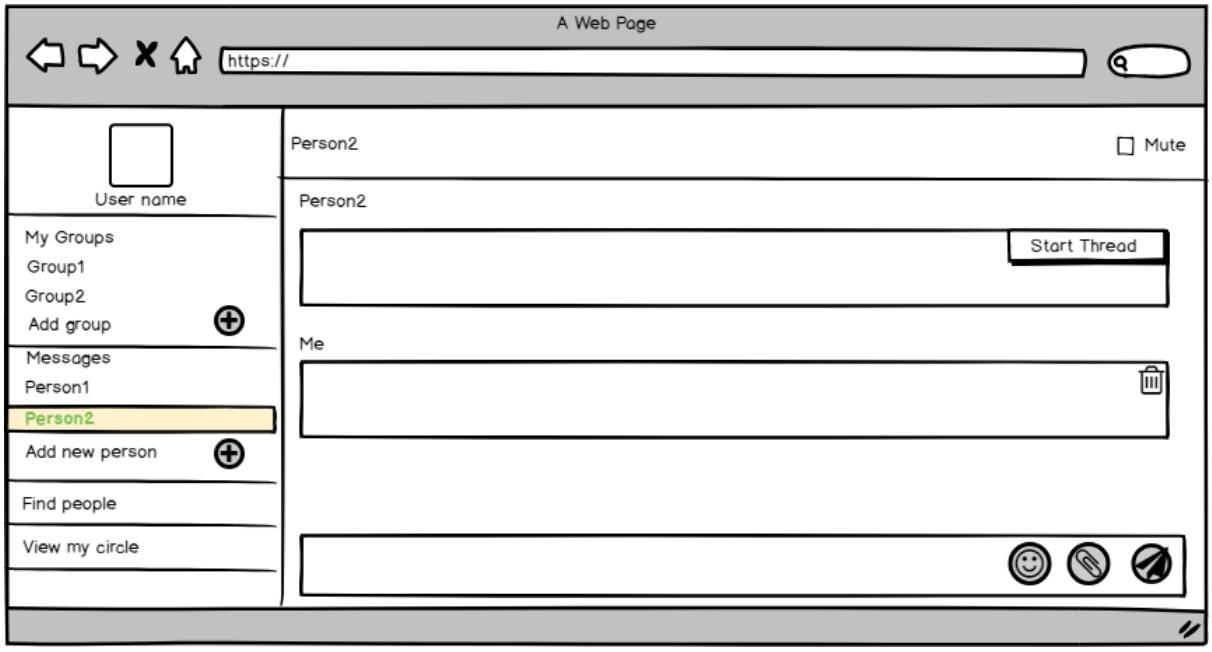


Post-login screen

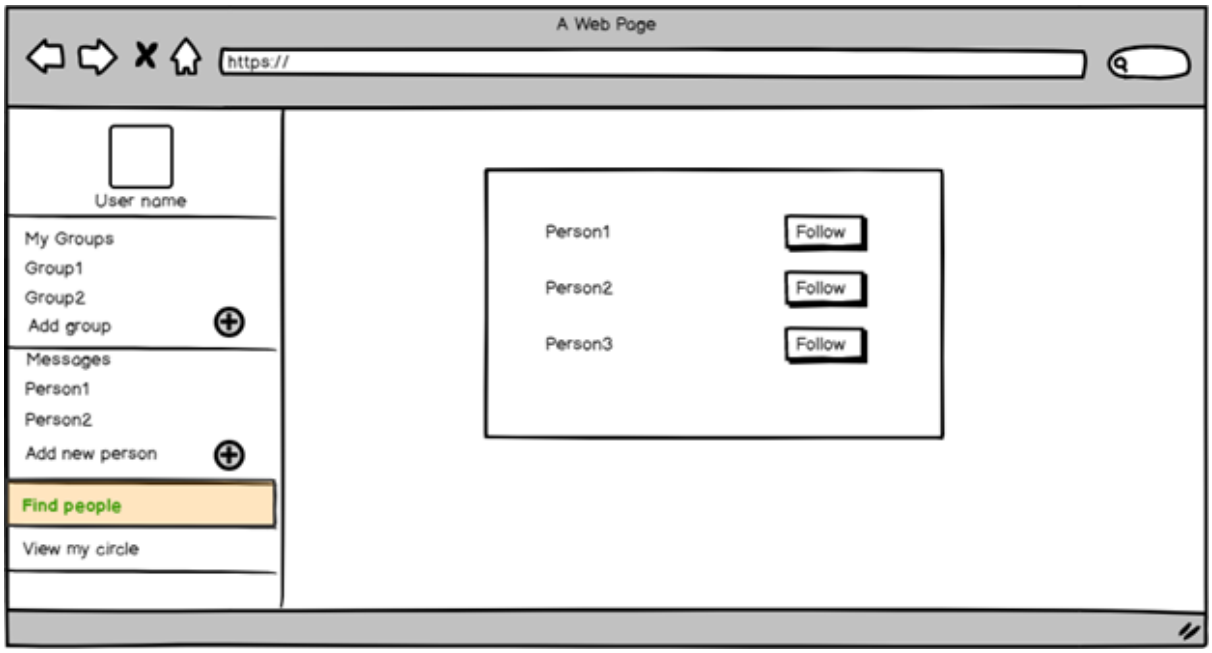
Group chat window



Personal chat window



Find people and follow



View circle