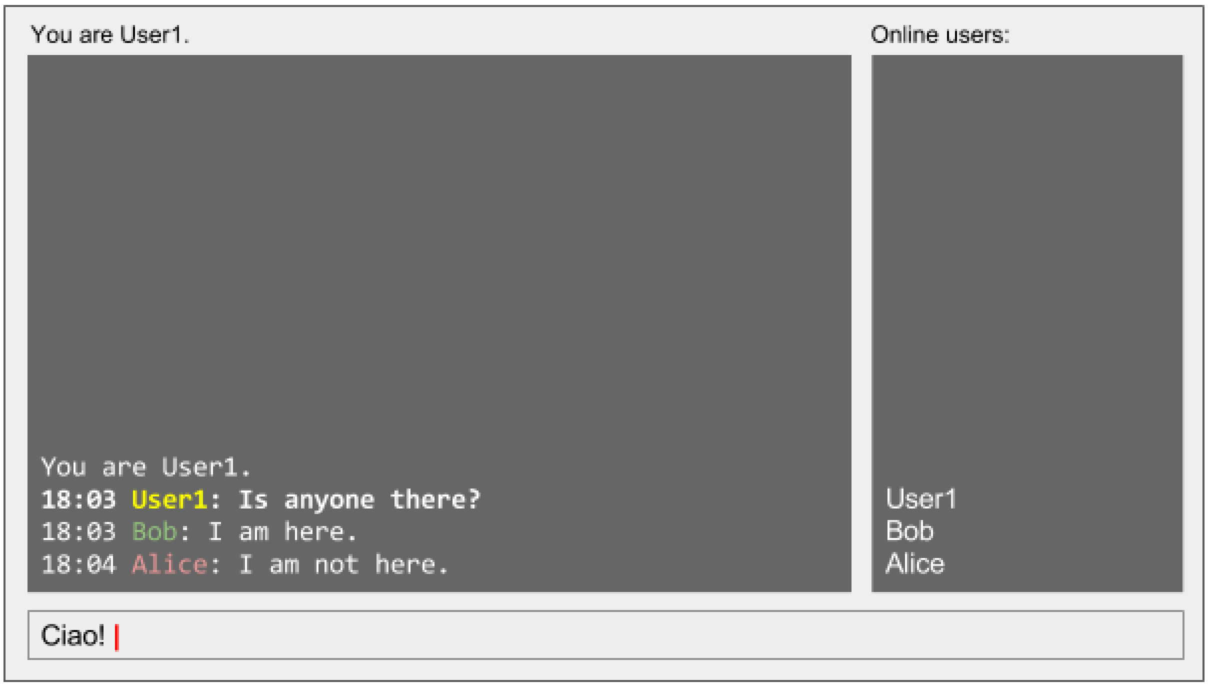
SENG 513 - Assignment 3

Due date: March 10, 2017 at 23:59pm

Weight: 15% of the final grade.

Note: This is an **individual** assignment. Group work is **NOT** allowed.

Implement a web application for online chatting. The user interface should look similar to the one below:



```
You are User1.
18:03 User1: Is anyone there?
18:03 Bob: I am here.
18:04 Alice: I am not here.
```

As a starting point for this assignment you can follow the tutorial posted at:

http://socket.io/get-started/chat/

The site hosting the tutorial above is not always running, so I have made the code available at:

https://github.com/uofc-seng513w17/chat-example

The above tutorial and the corresponding source code implements only very limited chat functionality. For this assignment you are asked to implement additional functionality, as described below.

You do not have to use websockets to implement this assignment. However, I strongly recommend you use websockets in combination with socket.io. Also, you **may use any** external JavaScript libraries for this assignment.

Requirements

All requirements below have the same weight, but not all are equally easy to implement. Please plan accordingly.

Presentation: / 10 points

- Your application should look attractive.
- If the browser is resized, your application should resize as well (ie. basic responsive design).
- You are free to implement more sophisticated responsive web design features, as described here: https://developers.google.com/web/fundamentals/design-and-ui/responsive/

Time stamps: / 10 points

- All chat messages should be displayed with a timestamp indicating when the message was sent.
- The timestamp should be calculated by the server.

Nicknames: / 10 points

- When a user navigates to the web application, the user should be assigned a unique username. This nickname should be generated by the server. The nickname should be somehow indicated to the user, for example by being displayed at the top of the page as suggested in the proposed layout.
- When the user types in a message, the message should be shown to everyone with the user's nickname.

Scroll-up text: / 10 points

- The chat log should be aligned at the bottom.
- The chat log should have a vertical scroll bar.

Chat log history: / 10 points

- The server should remember at least the last 200 messages. It is ok to store this history in memory rather than persisting it on disk.
- When a user connects to the system, the entire chat log should be displayed to the user.

Display current users: / 10 points

- The app should display all currently connected users. When a new user joins or an existing user disconnects, this list should be updated.

Change nickname: / 10 points

- Users should be able to change their nickname by typing a special command into the textbox:

  ```
  /nick <new nickname>
  ```

- This command should be refused if the nickname is not unique.
- After successful nickname change the list of currently logged in users should be updated.

Nick color: / 10 points

- Users should be able to change the color of their nickname using a command:

  ```
  /nickcolor RRGGBB
  ```

- Once the color has been changed, all subsequent messages from that user should be displayed with the new colored nickname.

Bold messages: / 10 points

- Messages posted by the user should be bolded in the chat log (or otherwise stylized).

Cookies: / 10 points

- If the user gets disconnected and then the user reconnects again, the user should be assigned the same nickname. You can use browser cookies for this purpose.

- You may find it easier to work with cookies using client-side JavaScript. Some information regarding this is available here: https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie

Additional requirements:

In order to receive any marks for this assignment, you also need to satisfy the following:

- Your chat application must support at least 5 concurrent users.
- Your implementation must work at least in Google Chrome **and** Mozilla Firefox browsers.
- Your implementation must work on the Linux machines in the labs on the main floor of MS.
- Your entire source code must be posted on Github in a public repository.

Submission:

Submit a readme.pdf file containing:

- name and ID;
- link to the GitHub repository containing your code;
- any special instructions on how to run the code on the Linux machines in the lab.

The instructions must outline the steps that the TA needs to perform in order to test your submission.

General information about all assignments:

1. **Due time** is listed at the top of the assignment**.** Late assignments or components of assignments will not be accepted for marking without your instructor's approval for an extension beforehand. What you have submitted in D2L as of the due date is what will be marked.
2. **Extensions** may be granted for reasonable cases, but only by the course instructor, and only with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Cases where extensions will not be granted include situations that are typical of student life, such as having multiple due dates, work commitments, etc. Forgetting to hand in your assignment on time is not a valid reason for getting an extension.
3. After you submit your work to D2L, make sure that you check the content of your submission. It's your responsibility to do this, so make sure that your submit your assignment with enough time before it is due so that you can double-check your upload, and possibly re-upload the assignment.
4. All assignments should include contact information, including full name, student ID and tutorial section, at the very top of each file submitted.
5. Assignments must reflect **individual** work. Group work is **not allowed** for assignments, nor can you copy the work of others. For further information on plagiarism, cheating and other academic misconduct, check the information at this link: http://www.ucalgary.ca/pubs/calendar/current/k-5.html.
6. You can and should submit many times before the due date. D2L will simply overwrite previous submissions with newer ones. It's better to submit incomplete work for a chance of getting partial marks, than not to submit anything.
7. Only one file can be submitted per assignment. If you need to submit multiple files, you can put them into a single container. The container types supported will be ZIP and TAR. No other formats will be accepted.
8. Assignments will be marked by your TAs. If you have questions about assignment marking, contact your TA first. If you still have question after you have talked to your TA then you can contact your instructor.