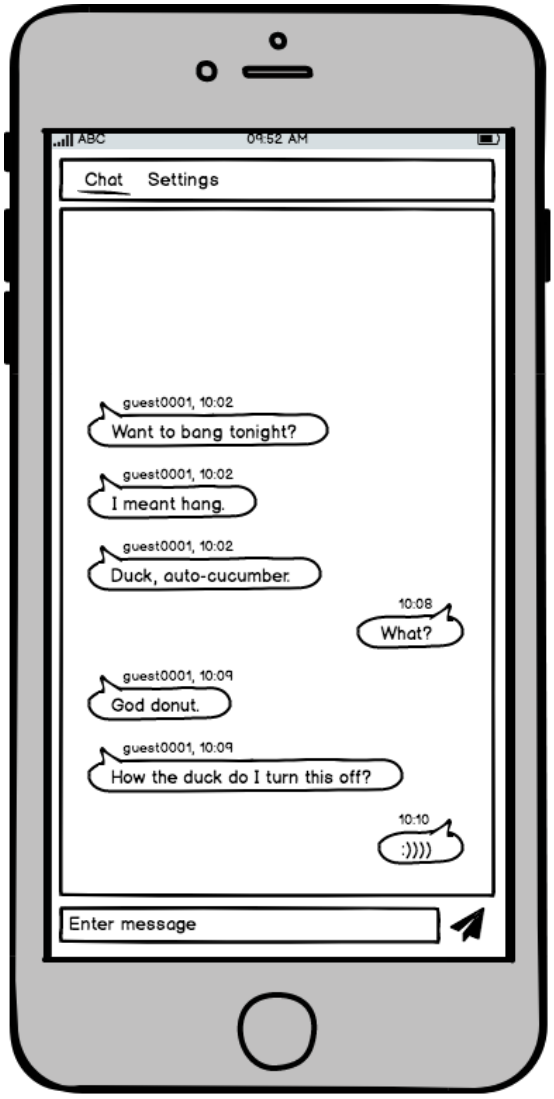
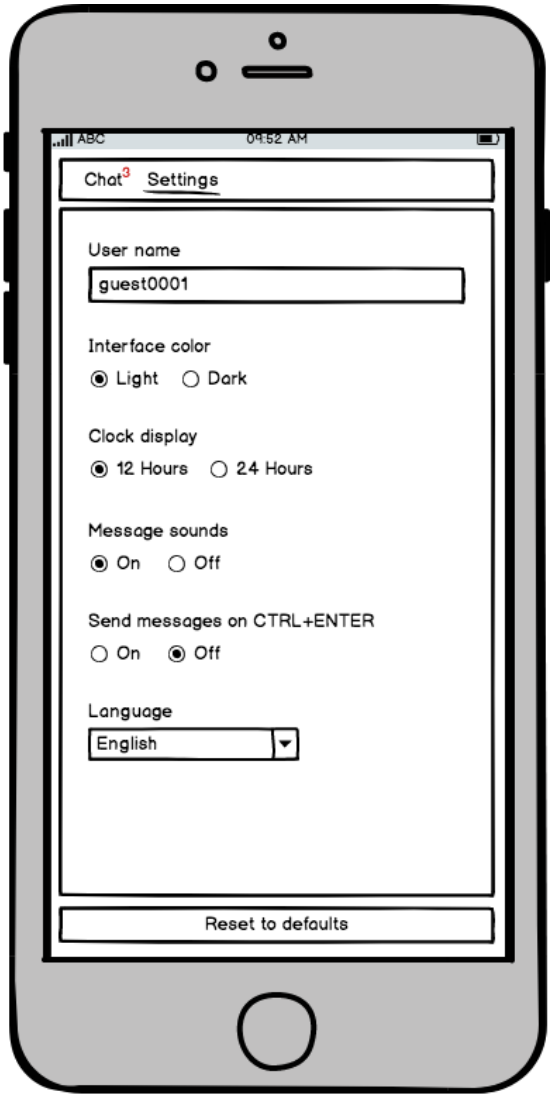
Frontend homework

Create a single page application based on following mockups:





Application should contain 2 pages:

1. Chat page
2. Settings page

Chat page

It's a socket.io based chat interface. When the user opens the application, he/she meets this view.

It contains a chat message box, where the sent messages are floating right and the received messages are floating left. The message contains the text of the message, date time (optional) and the user name if it is not the current user.

There is a text input field and a send button at the bottom of the page.

If the user is on another tab and he/she gets a message, the chat tab will blink, until he doesn't read the message.

Optional:

1. Smiles support
2. Unread messages count in the chat tab

3. Link parser
    a. Youtube link (embedded video should appear)
    b. Link to an image (embedded image should appear)
    c. All other links should appear as anchor
4. Any your own ideas are welcome

You should listen to event “message”, and you should trigger the event “message” also.

The example data is:

```
{ message: "Hi", user: "guest0001" }
```

* Socket.io server address: **http://185.13.90.140:8081/**

* Socket.io version: **0.9**

Settings page

User can modify the following settings:

1. User name
2. Interface color
    a. Light
    b. Dark
3. Clock Display
    a. 12 hours
    b. 24 hours

Optional:

1. Message sounds
    a. On
    b. Off
2. Send messages on CTRL+ENTER
    a. On
    b. Off
3. Internationalization (It’s enough to have just one additional language)
4. Any your own ideas are welcome

All settings should be saved in local storage and there must be a button “Reset to defaults”, by clicking on it application should fallback to default settings.

Requirements

1. You have to use one popular javascript framework.
2. You have to use css preprocessors.
3. It should work on every desktop and phone, so you have to make responsive design. And it has to work both portrait and landscape orientation.
4. We like the clean, commented, small and modularized codes.
5. Working code, that works if we serve it with the http server and open in a browser.
6. Readme file that contains what is it, how does it work and how could we setup.

Optional:

1. Unit testing.
2. Webpack.
3. JSdoc markdown.
4. CSS Modules.

Nice and intuitive design with good user experience is an extra point.