# Frontend task

Implement a web client for a Socket.IO chat server. You must, at a minimum, implement the following:

- A pseudo login screen, with username & password
- Emmitting and receiving of `message` events, described below
- Emmitting and receiving of `command` events, described below
- Displaying widgets in response to server `command` events
- Do your best in terms of UI and UX design
- Document the project so that another developer could start working with

Feel free to use whatever toolchain you're most comfortable with.

The chat server itself can be accessed at https://demo-chat-server.on.ag/

# Protocol

The server implements the following protocol over web sockets:

# Event types

The server can handle the following events:

## *message*

```
{
  author: "<username>",
  message: "<message>"
}
```

## Example:

### Message

```
{
  author: "Client",
  message: "Hello!"
}
```

### Response:

```
{
  author: "ottonova bot",
  message: "Hey Client, you said 'Hello!', right?"
}
```

## *command*

When the server receives an event of this type, it will ignore the payload and respond with a random command:

```
{
    author: "ottonova bot",
    command: {
      type: <type>,
      data: <data>
    }
}
```

Where `type` is one of the following:

## Date

```
{
  type: "data",
  data: "2018-01-01T14:32:33.921Z"
}
```

The goal of this command is to allow the user to easily pick a date in the future, starting and including the date supplied in the command message.

You must then, in response to this command render the following buttons on the screen:

- Monday, Tuesday, Wednesday, Thursday, Friday.

You must pick the correct starting day based on the supplied date, for example if the supplied date is Wednesday, the buttons must be in the order:

- Wednesday, Thursday, Friday, Monday, Tuesday

## Map

```
{
  type: "map",
  data: {
    lat: 48.1482933,
    lng: 11.586628
  }
}
```

The goal of this command is to show the user a location on a map.

You must then, in response to this message, draw a small map on the screen with a marker located at the specified co-ordinates.

## Rate

```
{
  type: "rate",
  data: [<i>, <j>]
}
```

where both `i` and `j` are integers and 0 < i < j <= 5.

The goal of this message is to allow the user to rate the conversation.

You must then, in response to this message, draw the appropriate of buttons on the screen.

## Complete

```
{
  type: "complete",
  data: ["Yes", "No"]
}
```

The goal of this message is to allow the user to close the conversation.

You must draw two buttons on the screen, with the corresponding labels.

## For all command cases

- For all cases you must display a message informing the user what the purpose of the widgets is. Ex: for “rate” you might display “Please rate your experience”, and then the buttons. This detail is left up to you.
- You must only allow a user to interact with a widget once.
- Once an option is selected by the user you must emit a `message` event the user’s choice.
- Once an widget’s role has been fulfilled, it must be hidden and replaced with the chosen response as if the user typed. Ex: If the following buttons appear: [Monday, Tuesday, Wednesday, Thursday, Friday] and the user clicks on “Tuesday”, all the buttons must be hidden and the message `<username> Tuesday` must appear on the screen.