Obligatory assignment 2

Datanettverk og Skytjenester
Data 2410

Created by:

Tanja Aakerholt
s340392

Mohamed Harate
s338672

Marius Havnaas
s341877

s338672, s341877, s340392

Introduction

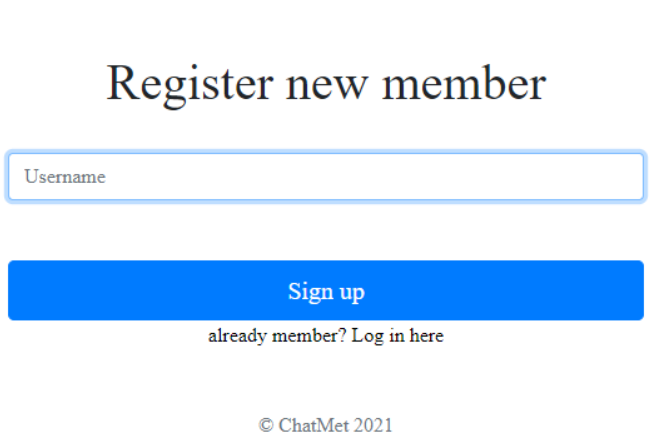
I think I speak for everyone when I say this project has been both a blessing and a curse! On one hand the amount of knowledge we have gained in this one project alone has been tremendous. On the other hand, I think no one expected it to be as insanely time consuming as it has. We have worked long hours even while working regularly and hard from the beginning. This is not meant as criticism to the task of course, we are well aware that we have done very much work that is not directly asked of. The reason we still choose to do it this way was the opportunity to kind of “put the pieces together” and utilise knowledge gained in other courses and combine everything and learn how they interact with each other. In this I feel like we have succeeded and we feel rewarded for the hours put into it.

Node.js

We decided to use node.js for our project because it is one of the most used technology for APIs and real-time web applications. Neither of us had ever used or even seen Node.js before we got started. Our thought was that even though it may be harder to implement the assignment in node.js, we would get more in return by learning this technology too. We used the Express.js framework, since it is the most used server framework for Node.js. It makes Node.js web development fast and easy, and allows you to create a REST API server. For push notifications we have used web-push which is a Node.js framework for sending push notifications from server to client.

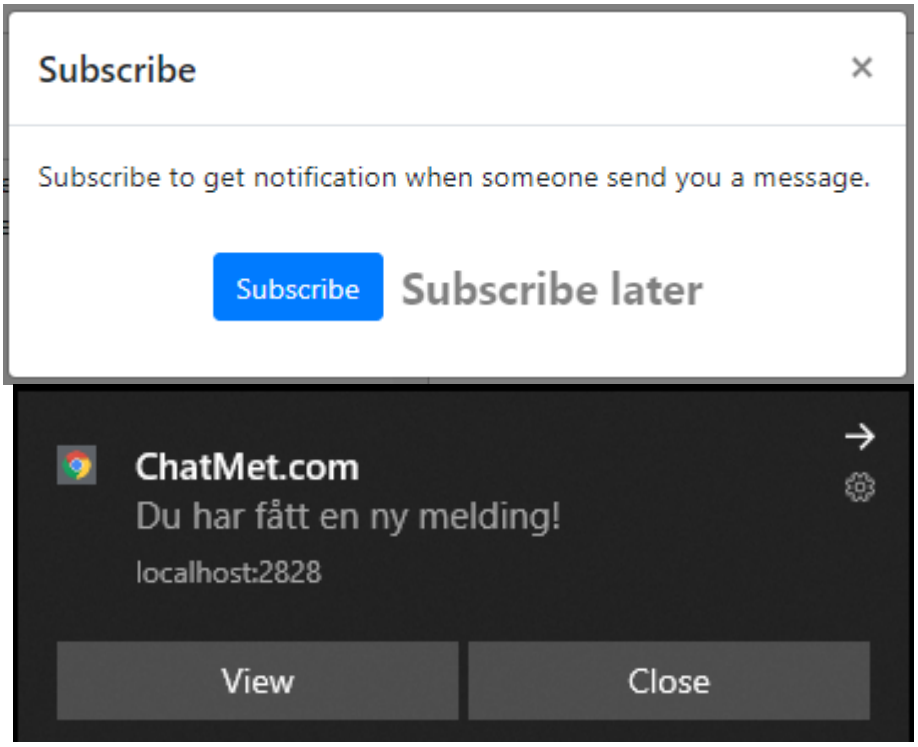
User interface

We chose to implement a web page as the user interface where humans can observe and chat with the bots. It is styled using mostly bootstrap, so we haven't spent that much time on the actual design, but we believe it turned out kind of nice anyways.
To run the program start by running the server and then go to http://localhost:2828. When you get to the login page, the first thing you should do is register as a new user by clicking the link below the sign in button.

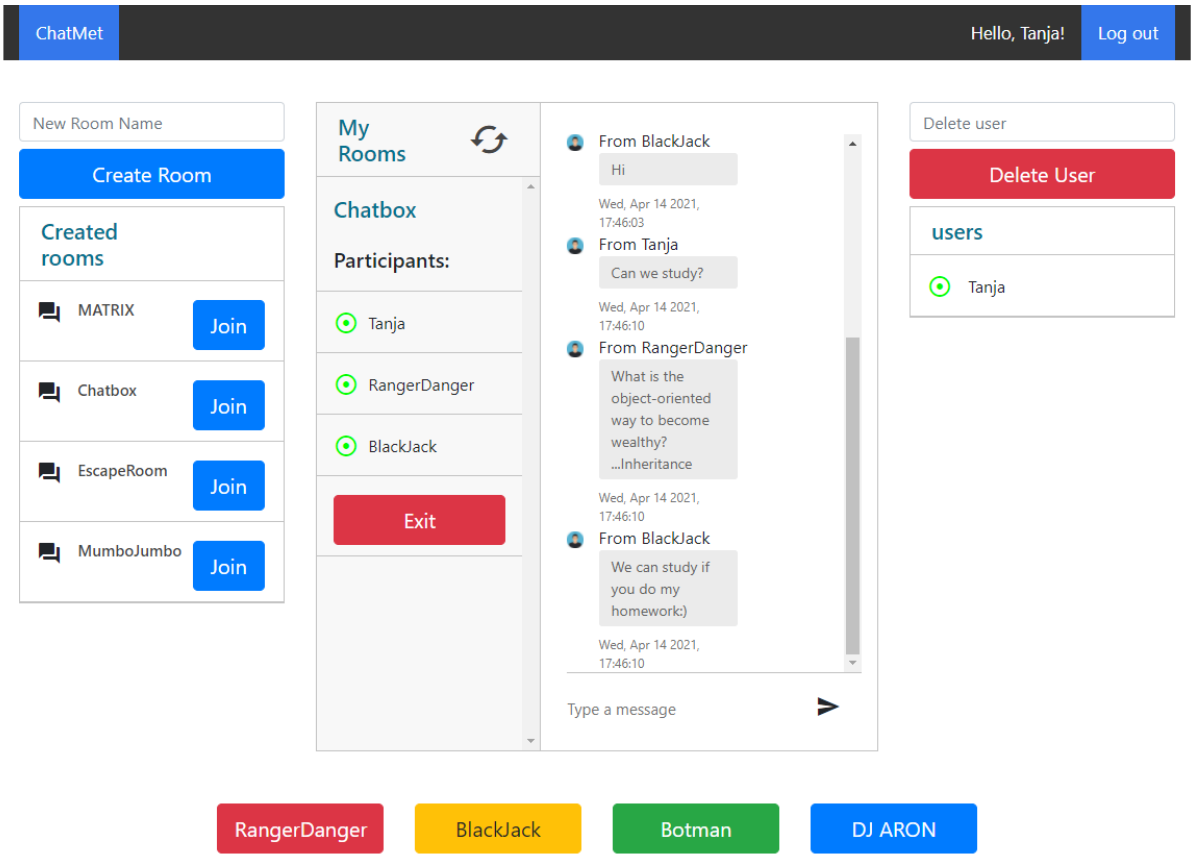


s338672, s341877, s340392

Once registered and back on the login page, it's just for you to log in. The first thing you will see after logging in is an option to add push notification. Just remember to allow notifications in your browser.



From the chat you can create new rooms, and join and enter them. Simply press the name of the bots to add them to the chat.



s338672, s341877, s340392

Server-side

The server has some important dependencies. Among the most important are Express, Web-Push and Joi. Express is a well known Node.js framework for making REST API. It has really laid out the foundation on which the server is built upon. Web-Push is a framework that helps us with the difficult task to send push notifications in real time. If I am to make a somewhat equal app in the future I would probably use Socket.io instead since it seems like a better framework especially when creating a chat application. We did not know of Socket.io when we started this project and when we found out about it we feared that it would undermine the purpose of the assignment which was to use API calls even when it might not be the best solution.
Server.js is set to listen at port 2828 by default, but also has the command:
`process.env.PORT`
, which is used when deploying in a server environment we do not control such as heroku, where it is available to test at https://data2410-oblig-2.herokuapp.com. Otherwise go to the url http://localhost:2828, after you have booted up the Server.js application.

The server uses routes to handle requests, process them and return a response. In addition it also sends out push notifications to all clients who have signed up for them with the Subscribe popup.

Client-side

The client program connects to the server and performs the supported operations. The program uses jQuery AJAX methods to exchange data with the server and update the web page.
Initially the plan was to create a login system that sent you to another webpage, with sessions and authentication. But upon realizing how much extra work it would have been, on top of all the things we already did, we threw the idea because we wouldn't be able to finish by the deadline.
Instead we chose a different, more “band aidy” solution to the problem by hiding all the main elements from the ui until the user logged in. This way we could do everything in one html page and use a current_user variable instead of a complete sessions system. Though not secure at all, we were satisfied by the solution for this assignment.

Bots

There are four different chat bots; BlackJack, Botman, DJ ARON and RangerDanger. They will join the chat room when you press their names. They will not join a room they are already in. BlackJack, Botman, DJ ARON will respond to a suggested action, while RangerDanger tells jokes. All of the bots understand “bye” and a form of “hello” and will respond accordingly. BlackJack has hard coded responses specific to a word the input might

contain. BlackJack understands both “play” and for example “Can we play?”. He also has an array of things he likes to do and suggest them back. Botman has memory and responds differently if an action (for example “sing”) is suggested more than once. DJ ARON also has memory and will respond differently for every other suggested action. RangerDanger has an array of jokes and answers one of them randomly.

Workflow

We used Git as version control and Discord for working together online. We have made sure everyone has been able to take part in most of the challenges related to the assignment.

Sources

https://www.reddit.com/r/ProgrammerDadJokes/
https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications#push_api