**Back-end of the search bar:**

a search feature was added to search for services. The search key words are meant for only the title of the service (not the description). The search is not case-sensitive, the user doesn't have to type the exact words to match up to a title but he can write a small keyword and the server will find out if this keyword is included in the title of the service.

e.g.: if a user type the word "shoes" the server will output all the services that have "shoes" in the title or more than this word.

The search route is found in the services routes (/routes/services.js):

// @route GET api/service/search

// @desc search for a service

// @access Private

// @errors noservice error

router.get(

'/search',

passport.authenticate('jwt', { session: false }),

serviceController.search

);

The search controller is found in the services controller (/controllers/serviceController/search.js):

const mongoose = require('mongoose');

// Models

const Service = mongoose.model('service');

module.exports = search = (req, res) => {

const errors = {};

Service.find({ name: new RegExp('^' + req.params.name + '$', 'i') })

.sort({ date: -1 })

.then((services) => {

if (services.length === 0) {

errors.noservices = 'No services found';

return res.status(404).json(errors);

}

return res.json(services);

})

.catch((err) => {

errors.error = 'Error fetching services from database';

res.status(500).json({ ...errors, ...err });

});

};

**Back-end of the chatting system:**

The chatting system is established between two users only, it's a private chat between the service owner and the service helper.

const express = require('express');

const router = express.Router();

const passport = require('passport');

const ChatController = require('../controllers/chatController/index');

// @route POST api/profile/id1+id2

// @desc Open chat between 2 users

// @access Private

// @errors noprofiles error

router.post(

'/:id1+:id2',

passport.authenticate('jwt', { session: false }),

ChatController.openChat

);

module.exports = router;

We used socket.io package to take information from the client and send it to the server then the server distributes the message to the client on the other end.

const io = socket(server);

io.on('connection', function(socket){ // waiting for connection with a client

console.log('socket connection made and the socket id is ' + socket.id);

socket.on('chat', function(data){

// waiting for a data to be send from a client

io.sockets.emit('chat', data)

// send the message back from the server to the rest of the clients

});

});

The chat history will be stored in mongoDB database and it consists of first user id, second user id, message of the first user and message of the second user.

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const ChatSchema = new Schema({

userID1: {

type: mongoose.Schema.Types.ObjectId,

ref: 'user',

required: true

},

msg1: {

type: String,

required: true

},

userID2: {

type: mongoose.Schema.Types.ObjectId,

ref: 'user',

required: true

},

msg2: {

type: String,

required: true

}

});

const Chat = mongoose.model('Chat', ChatSchema);

module.exports = Chat;