

## 27. Understanding Websockets & Socket.io

### \* Chapter 400: Module Introduction





#### What's In This Module?

Why Realtime?

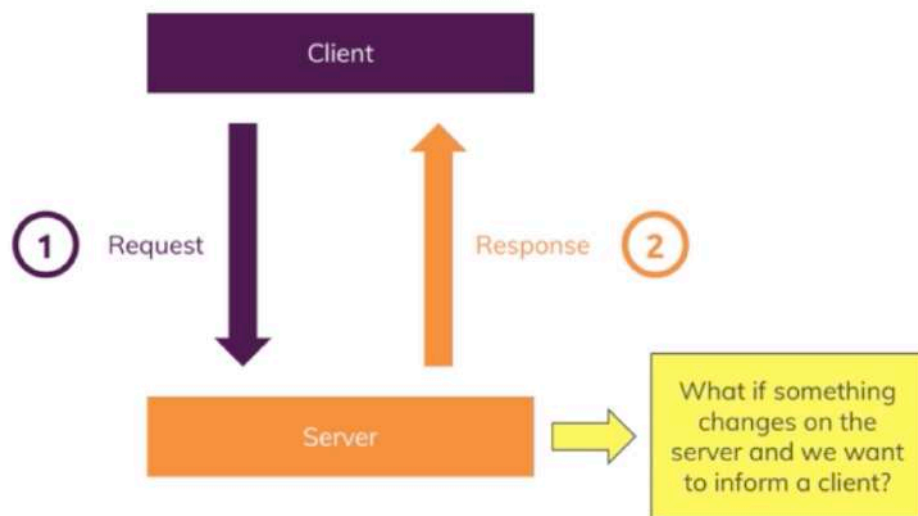
How to Add Realtime Communication to a Node App

### \* Chapter 401: What Are Websockets & Why Would You Use Them?

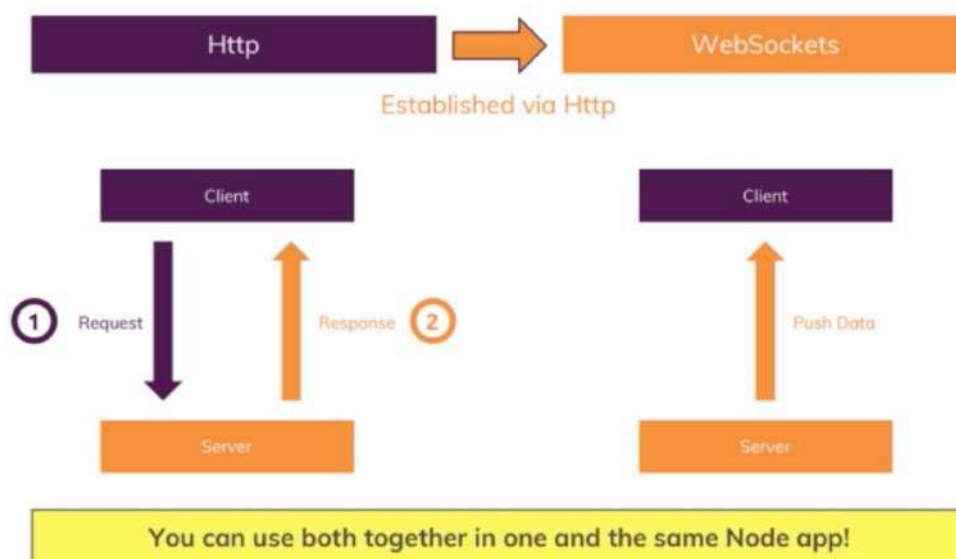




## How It Currently Works



## WebSockets instead of Http



- what if something changed on the server and we actively wanna inform a client well then we can use websockets instead of HTTP.

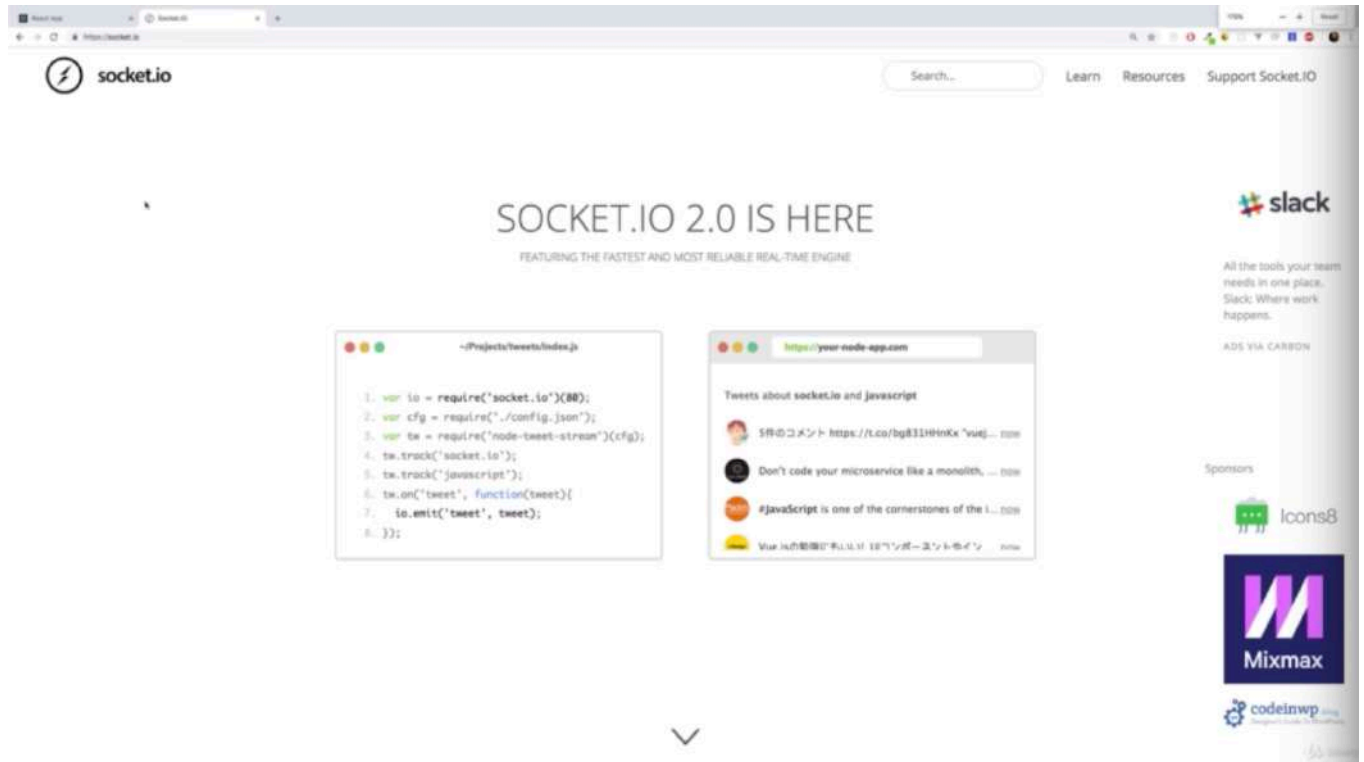
- websockets built up on HTTP. they are established via HTTP. they use a so-called HTTP handshake to upgrade the HTTP protocol to the Websockets protocol and the websockets protocol that simply talks about how data is exchanged. protocol is something you don't have to manage. the browser and the server communicate through a protocol and protocol define how the communication can happen. HTTP it's request response. with websockets, it's push data are it's both. we can also send data from the client to the server does is still included.

- we can push data from the server to declined and you can and you will use both together in one in the same nodes. so it's not like you have to decide do i build an app with websockets or do i build one with HTTP.

## \* Chapter 402: Websocket Solutions - An

# Overview



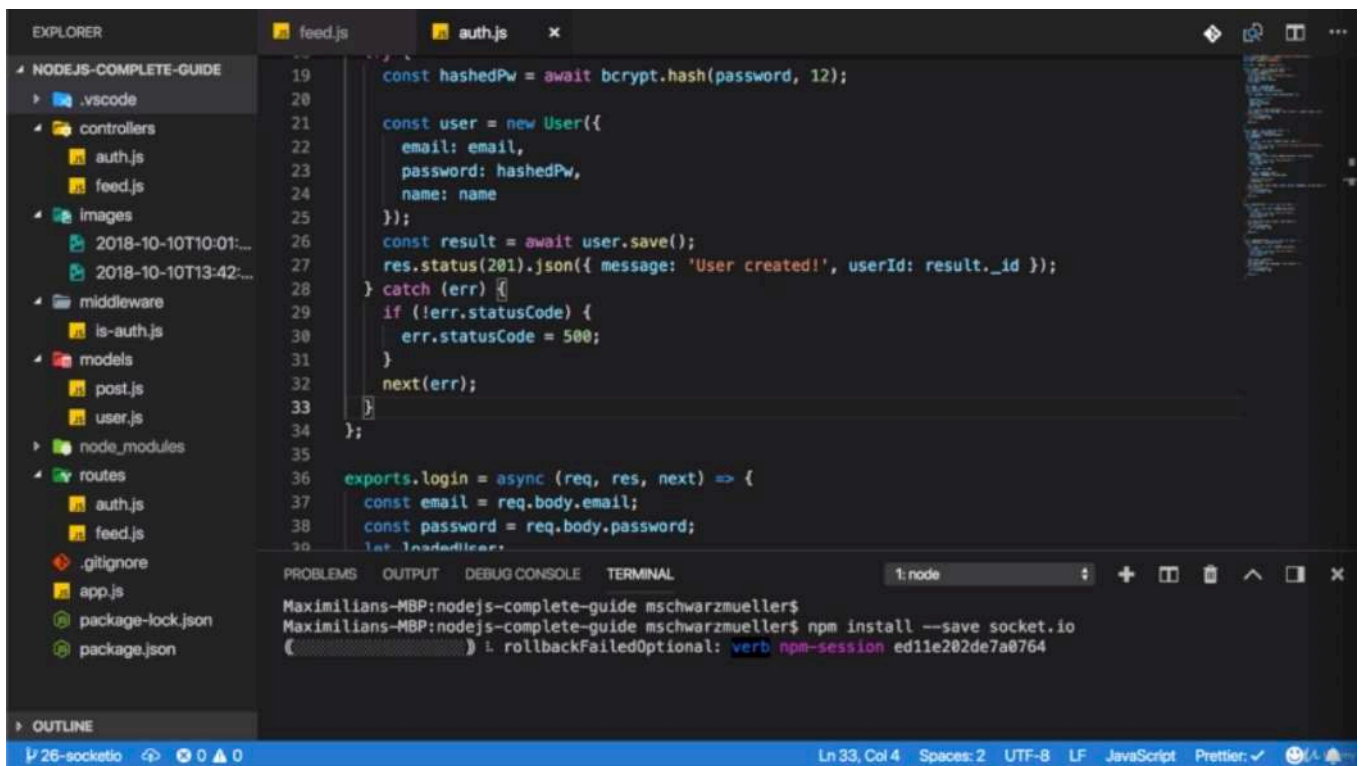


- one of the most popular packages is 'socket.io' which is the package that uses websockets.

## \* Chapter 403: Setting Up [Socket.io](https://socket.io/) On The Server

- 1. update
- app.js(b)





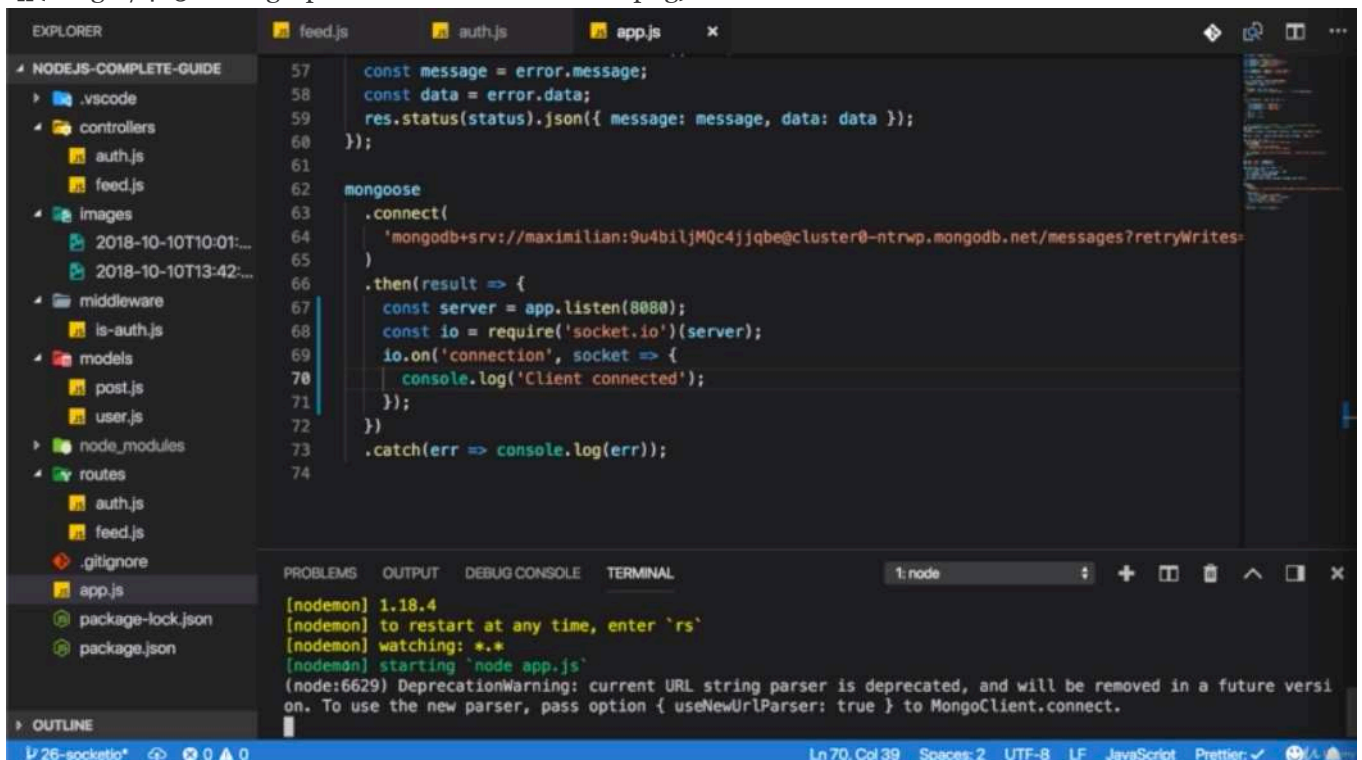
```
19 const hashedPw = await bcrypt.hash(password, 12);
20
21 const user = new User({
22   email: email,
23   password: hashedPw,
24   name: name
25 });
26 const result = await user.save();
27 res.status(201).json({ message: 'User created!', userId: result._id });
28 } catch (err) {
29   if (!err.statusCode) {
30     err.statusCode = 500;
31   }
32   next(err);
33 }
34 };
35
36 exports.login = async (req, res, next) => {
37   const email = req.body.email;
38   const password = req.body.password;
39   let loadedUser;
```

Maximilians-MBP:nodejs-complete-guide mschwarzmuellers  
Maximilians-MBP:nodejs-complete-guide mschwarzmuellers\$ npm install --save socket.io  
(node:6629) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

- we have to add [socket.io](https://socket.io/) on both the server and the client. so both on the node app and on the react app because client server will communicate through web sockets. so we have to establish that communication channel on both ends on the front end react and on the backend node.

- 'npm install --save socket.io' on the backend





```
57 const message = error.message;
58 const data = error.data;
59 res.status(status).json({ message: message, data: data });
60 });
61
62 mongoose
63 .connect(
64   'mongodb+srv://maximilian:9u4biljMQc4jjqbe@cluster0-ntrwp.mongodb.net/messages?retryWrites=
65 )
66 .then(result => {
67   const server = app.listen(8080);
68   const io = require('socket.io')(server);
69   io.on('connection', socket => {
70     console.log('Client connected');
71   });
72 })
73 .catch(err => console.log(err));
74
```

[nodemon] 1.18.4  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: \*.\*  
[nodemon] starting `node app.js`  
(node:6629) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

- if i run npm start, will never see 'client connected' in the console because we established all that on the server side. we not got a waiting socket connection or port. but we got no client which would connect and that is the next step we have to

```
1 //app.js(b)
2
3 const path = require('path');
4
```

```

5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const multer = require('multer');
9
10 const feedRoutes = require('./routes/feed');
11 const authRoutes = require('./routes/auth');
12
13 const app = express();
14
15 const fileStorage = multer.diskStorage({
16   destination: (req, file, cb) => {
17     cb(null, 'images');
18   },
19   filename: (req, file, cb) => {
20     cb(null, new Date().toISOString() + '-' + file.originalname);
21   }
22 });
23
24 const fileFilter = (req, file, cb) => {
25   if(
26     file.mimetype === 'image/png' ||
27     file.mimetype === 'image/jpg' ||
28     file.mimetype === 'image/jpeg'
29   ) {
30     cb(null, true);
31   } else {
32     cb(null, false);
33   }
34 }
35
36 // app.use(bodyParser.urlencoded()); // x-www-form-urlencoded <form>
37 app.use(bodyParser.json()); // application/json
38 app.use(multer({storage: fileStorage, fileFilter: fileFilter}).single('image'))
39 app.use('/images', express.static(path.join(__dirname, 'images')));
40
41 app.use((req, res, next) => {
42   res.setHeader('Access-Control-Allow-Origin', '*');
43   res.setHeader(
44     'Access-Control-Allow-Methods',
45     'OPTIONS, GET, POST, PUT, PATCH, DELETE'
46   );
47   res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
48   next();
49 });
50
51 /**app.js(b) file is the first file that runs
52 * when we start our server
53 * and there we should set up or socket.io connections we wanna expose
54 * as we set up our routes there.
55 * in the end we forward requests to our routes
56 * and you could have all the socket.io set up in a different file
57 * but i will do it here.
58 *
59 * as we set up our routes for the normal HTTP requests
60 * we can set up our socket.io channels

```

```

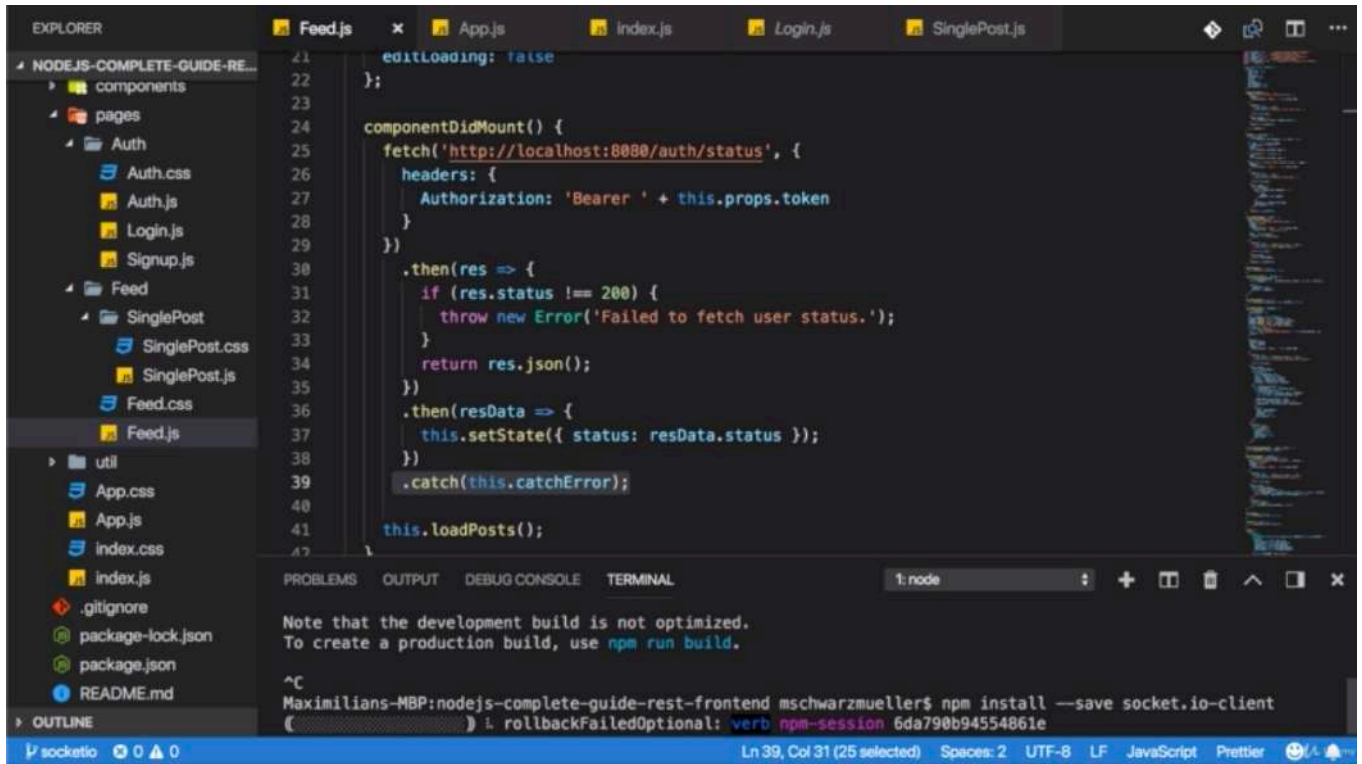
61 * and keep it mind that socket.io uses a different protocol
62 * websockets and there websocket requests will not interfere with the normal HTTP requests
63 * which are sent by default by the browser.
64 */
65 app.use('/feed', feedRoutes);
66 app.use('/auth', authRoutes);
67
68 app.use((error, req, res, next) => {
69   console.log(error);
70   const status = error.statusCode || 500;
71   const message = error.message;
72   const data = error.data;
73   res.status(status).json({ message: message, data: data });
74 });
75
76 mongoose
77   .connect(
78     'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/message?retryWrites=true'
79   )
80   .then(result => {
81     /**'listen()' method does return us a new node server
82      * so we can store that in a constant.
83      * this is the node server we spun up.
84      */
85     const server = app.listen(8080);
86     /**this sets up socket.io
87      * and i mentioned that websockets built up on HTTP
88      * since this server uses HTTP,
89      * we used that HTTP server to establish our websocket connection
90      * that uses HTTP protocol as a basis.
91      */
92     const io = require('socket.io')(server);
93     /**we can use it to define a couple of event listeners for example.
94      * for example to wait for new connections
95      * so whatever in you client connects to us.
96      *
97      * and then we execute a certain function
98      * where we get the cilent the so-called 'socket'
99      * did connect as an argument or the connection as an argument to be precise
100     * so this is the connection between ourserver and the client
101     * which connected and this function will be executed for every new client that connects
102     * so not only one time but as often as required as many client as connect.
103     */
104     io.on('connection', socket => {
105       console.log('Client connected')
106     })
107   })
108   .catch(err => console.log(err));

```

## \* Chapter 404: Establishing A Connection From The Client

1. update
- ./src/pages/Feed/Feed.js(f)

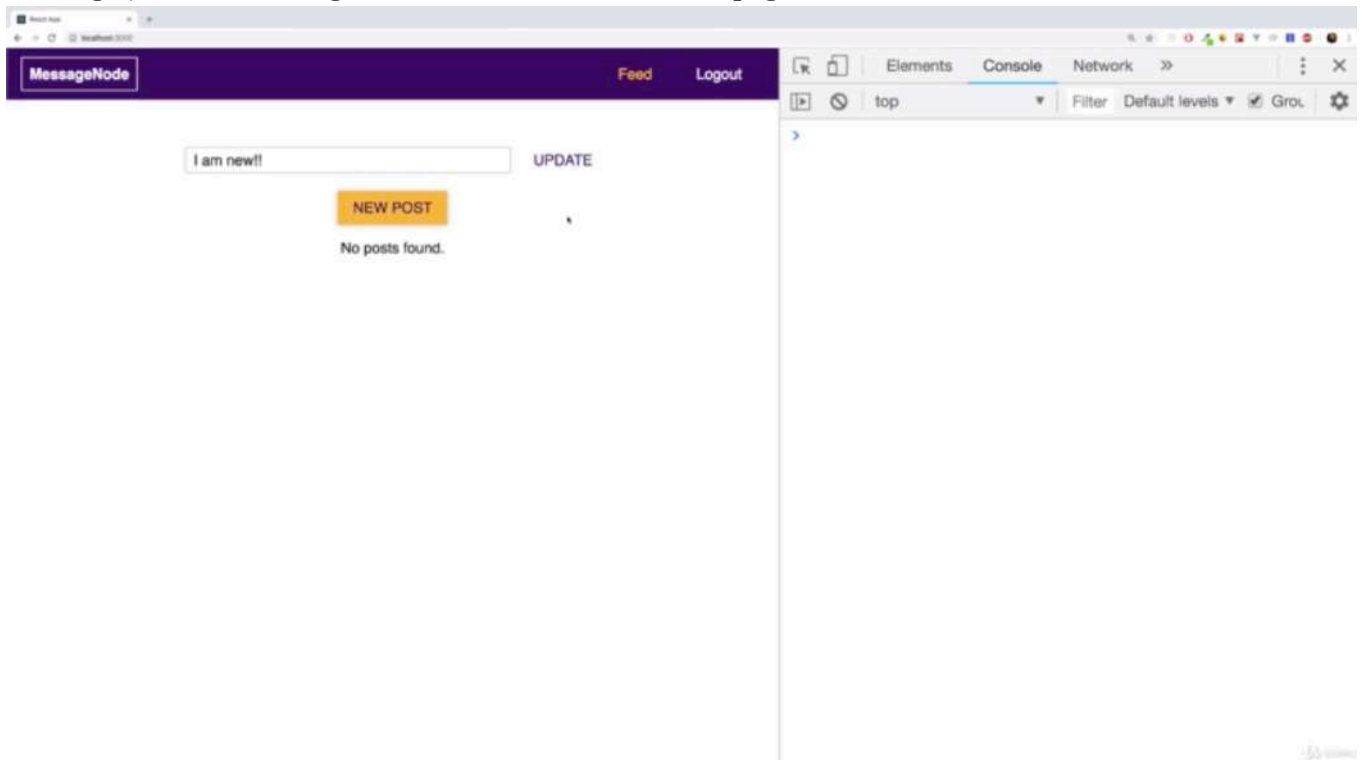




- 'npm install --save [socket.io](https://socket.io/)-client'

- so it's different package name it's [socket.io](https://socket.io/)-client because it's to code that will run on the client. this enter and this will in stall that package into our React project.





- this should have already executed this openSocket code because the page did reload.





```
57 const message = error.message;
58 const data = error.data;
59 res.status(status).json({ message: message, data: data });
60 });
61
62 mongoose
63 .connect(
64   'mongodb+srv://maximilian:9u4biljMQc4jjqbe@cluster0-ntrwp.mongodb.net/messages?retryWrites=
65 )
66 .then(result => {
67   const server = app.listen(8080);
68   const io = require('socket.io')(server);
69   io.on('connection', socket => {
70     console.log('Client connected');
71   });
72 })
73 .catch(err => console.log(err));
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
(node:6629) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
Client connected
```

- and if you go to the backend, you see 'client connected' because now we got an established connection between a client and our backend application through [socket.io](https://socket.io/) and all the other normal HTTP requests will still work as before

```
1  ../src/pages/Feed/Feed.js(f)
2
3  import React, { Component, Fragment } from 'react';
4  /**this will be a function
5   * that opens a new connection using socket
6   */
7  import openSocket from 'socket.io-client';
8
9  import Post from '../components/Feed/Post/Post';
10 import Button from '../components/Button/Button';
11 import FeedEdit from '../components/Feed/FeedEdit/FeedEdit';
12 import Input from '../components/Form/Input/Input';
13 import Paginator from '../components/Paginator/Paginator';
14 import Loader from '../components/Loader/Loader';
15 import ErrorHandler from '../components/ErrorHandler/ErrorHandler';
16 import './Feed.css';
17
18 class Feed extends Component {
19   state = {
20     isEditing: false,
21     posts: [],
22     totalPosts: 0,
23     editPost: null,
24     status: '',
25     postPage: 1,
26     postsLoading: true,
27     editLoading: false
28   };
29
30   componentDidMount() {
31     fetch('URL')
```



```

32 .then(res => {
33   if (res.status !== 200) {
34     throw new Error('Failed to fetch user status.');
```

35 }

```

36   return res.json();
37 })
38 .then(resData => {
39   this.setState({ status: resData.status });
40 })
41 .catch(this.catchError);
42
43 this.loadPosts();
44 openSocket('http://localhost:8080');
45 }
46
47 loadPosts = direction => {
48   if (direction) {
49     this.setState({ postsLoading: true, posts: [] });
50   }
51   let page = this.state.postPage;
52   if (direction === 'next') {
53     page++;
54     this.setState({ postPage: page });
55   }
56   if (direction === 'previous') {
57     page--;
58     this.setState({ postPage: page });
59   }
60   fetch('http://localhost:8080/feed/posts?page=' + page, {
61     headers: {
62       Authorization: 'Bearer ' + this.props.token
63     }
64   })
65   .then(res => {
66     if (res.status !== 200) {
67       throw new Error('Failed to fetch posts.');
```

68 }

```

69     return res.json();
70   })
71   .then(resData => {
72     this.setState({
73       posts: resData.posts.map(post => {
74         return {
75           ...post,
76           imagePath: post.imageUrl
77         };
78       }),
79       totalPosts: resData.totalItems,
80       postsLoading: false
81     });
82   })
83   .catch(this.catchError);
84 };
85
86 statusUpdateHandler = event => {
87   event.preventDefault();

```

```

88     fetch('URL')
89     .then(res => {
90         if (res.status !== 200 && res.status !== 201) {
91             throw new Error("Can't update status!");
92         }
93         return res.json();
94     })
95     .then(resData => {
96         console.log(resData);
97     })
98     .catch(this.catchError);
99 };
100
101 newPostHandler = () => {
102     this.setState({ isEditing: true });
103 };
104
105 startEditPostHandler = postId => {
106     this.setState(prevState => {
107         const loadedPost = { ...prevState.posts.find(p => p._id === postId) };
108
109         return {
110             isEditing: true,
111             editPost: loadedPost
112         };
113     });
114 };
115
116 cancelEditHandler = () => {
117     this.setState({ isEditing: false, editPost: null });
118 };
119
120 finishEditHandler = postData => {
121     this.setState({
122         editLoading: true
123     });
124     const formData = new FormData();
125     formData.append('title', postData.title);
126     formData.append('content', postData.content);
127     formData.append('image', postData.image);
128     let url = 'http://localhost:8080/feed/post';
129     let method = 'POST';
130     if (this.state.editPost) {
131         url = 'http://localhost:8080/feed/post/' + this.state.editPost._id;
132         method = 'PUT';
133     }
134
135     fetch(url, {
136         method: method,
137         body: formData,
138         headers: {
139             Authorization: 'Bearer ' + this.props.token
140         }
141     })
142     .then(res => {
143         if (res.status !== 200 && res.status !== 201) {

```

```

144     throw new Error('Creating or editing a post failed!');
145   }
146   return res.json();
147 })
148 .then(resData => {
149   console.log(resData);
150   const post = {
151     _id: resData.post._id,
152     title: resData.post.title,
153     content: resData.post.content,
154     creator: resData.post.creator,
155     createdAt: resData.post.createdAt
156   };
157   this.setState(prevState => {
158     let updatedPosts = [...prevState.posts];
159     if (prevState.editPost) {
160       const postIndex = prevState.posts.findIndex(
161         p => p._id === prevState.editPost._id
162       );
163       updatedPosts[postIndex] = post;
164     } else if (prevState.posts.length < 2) {
165       updatedPosts = prevState.posts.concat(post);
166     }
167     return {
168       posts: updatedPosts,
169       isEditing: false,
170       editPost: null,
171       editLoading: false
172     };
173   });
174 })
175 .catch(err => {
176   console.log(err);
177   this.setState({
178     isEditing: false,
179     editPost: null,
180     editLoading: false,
181     error: err
182   });
183 });
184 };
185
186 statusInputChangeHandler = (input, value) => {
187   this.setState({ status: value });
188 };
189
190 deletePostHandler = postId => {
191   this.setState({ postsLoading: true });
192   fetch('http://localhost:8080/feed/post/' + postId, {
193     method: 'DELETE',
194     headers: {
195       Authorization: 'Bearer ' + this.props.token
196     }
197   })
198   .then(res => {
199     if (res.status !== 200 && res.status !== 201) {

```

```

200         throw new Error('Deleting a post failed!');
201     }
202     return res.json();
203 })
204 .then(resData => {
205     console.log(resData);
206     this.setState(prevState => {
207         const updatedPosts = prevState.posts.filter(p => p._id !== postId);
208         return { posts: updatedPosts, postsLoading: false };
209     });
210 })
211 .catch(err => {
212     console.log(err);
213     this.setState({ postsLoading: false });
214 });
215 };
216
217 errorHandler = () => {
218     this.setState({ error: null });
219 };
220
221 catchError = error => {
222     this.setState({ error: error });
223 };
224
225 render() {
226     return (
227         <Fragment>
228             <ErrorHandler error={this.state.error} onHandle={this.errorHandler} />
229             <FeedEdit
230                 editing={this.state.isEditing}
231                 selectedPost={this.state.editPost}
232                 loading={this.state.editLoading}
233                 onCancelEdit={this.cancelEditHandler}
234                 onFinishEdit={this.finishEditHandler}
235             />
236             <section className="feed__status">
237                 <form onSubmit={this.statusUpdateHandler}>
238                     <Input
239                         type="text"
240                         placeholder="Your status"
241                         control="input"
242                         onChange={this.statusInputChangeHandler}
243                         value={this.state.status}
244                     />
245                     <Button mode="flat" type="submit">
246                         Update
247                     </Button>
248                 </form>
249             </section>
250             <section className="feed__control">
251                 <Button mode="raised" design="accent" onClick={this.newPostHandler}>
252                     New Post
253                 </Button>
254             </section>
255             <section className="feed">

```

```

256 {this.state.postsLoading && (
257   <div style={{ textAlign: 'center', marginTop: '2rem' }}>
258     <Loader />
259   </div>
260 )}
261 {this.state.posts.length <= 0 && !this.state.postsLoading ? (
262   <p style={{ textAlign: 'center' }}>No posts found.</p>
263 ) : null}
264 {!this.state.postsLoading && (
265   <Paginator
266     onPrevious={this.loadPosts.bind(this, 'previous')}
267     onNext={this.loadPosts.bind(this, 'next')}
268     lastPage={Math.ceil(this.state.totalPosts / 2)}
269     currentPage={this.state.postPage}
270   >
271     {this.state.posts.map(post => (
272       <Post
273         key={post._id}
274         id={post._id}
275         author={post.creator}
276         date={new Date(post.createdAt).toLocaleDateString('en-US')}
277         title={post.title}
278         image={post.imageUrl}
279         content={post.content}
280         onStartEdit={this.startEditPostHandler.bind(this, post._id)}
281         onDelete={this.deletePostHandler.bind(this, post._id)}
282       />
283     ))}
284   </Paginator>
285 )}
286 </section>
287 </Fragment>
288 );
289 }
290 }
291
292 export default Feed;
293

```

## \* Chapter 405: Identifying Realtime Potential

1. update

- ./src/pages/Feed/Feed.js(f)
- ./controllers/feed.js(b)

```

1  //./src/pages/Feed/Feed.js(f)
2
3  import React, { Component, Fragment } from 'react';
4  import openSocket from 'socket.io-client';
5
6  import Post from '../components/Feed/Post/Post';
7  import Button from '../components/Button/Button';
8  import FeedEdit from '../components/Feed/FeedEdit/FeedEdit';
9  import Input from '../components/Form/Input/Input';
10 import Paginator from '../components/Paginator/Paginator';
11 import Loader from '../components/Loader/Loader';

```

```

12 import ErrorHandler from '../components/ErrorHandler/ErrorHandler';
13 import './Feed.css';
14
15 class Feed extends Component {
16   state = {
17     isEditing: false,
18     posts: [],
19     totalPosts: 0,
20     editPost: null,
21     status: '',
22     postPage: 1,
23     postsLoading: true,
24     editLoading: false
25   };
26
27   componentDidMount() {
28     fetch('URL')
29       .then(res => {
30         if (res.status !== 200) {
31           throw new Error('Failed to fetch user status.');

```

```

68     if (direction === 'next') {
69         page++;
70         this.setState({ postPage: page });
71     }
72     if (direction === 'previous') {
73         page--;
74         this.setState({ postPage: page });
75     }
76     fetch('http://localhost:8080/feed/posts?page=' + page, {
77         headers: {
78             Authorization: 'Bearer ' + this.props.token
79         }
80     })
81     .then(res => {
82         if (res.status !== 200) {
83             throw new Error('Failed to fetch posts.');
```

```

84         }
85         return res.json();
86     })
87     .then(resData => {
88         this.setState({
89             posts: resData.posts.map(post => {
90                 return {
91                     ...post,
92                     imagePath: post.imageUrl
93                 };
94             }),
95             totalPosts: resData.totalItems,
96             postsLoading: false
97         });
98     })
99     .catch(this.catchError);
100 };
101
102 statusUpdateHandler = event => {
103     event.preventDefault();
104     fetch('URL')
105     .then(res => {
106         if (res.status !== 200 && res.status !== 201) {
107             throw new Error("Can't update status!");
108         }
109         return res.json();
110     })
111     .then(resData => {
112         console.log(resData);
113     })
114     .catch(this.catchError);
115 };
116
117 newPostHandler = () => {
118     this.setState({ isEditing: true });
119 };
120
121 startEditPostHandler = postId => {
122     this.setState(prevState => {
123         const loadedPost = { ...prevState.posts.find(p => p._id === postId) };

```



```

124
125     return {
126         isEditing: true,
127         editPost: loadedPost
128     };
129 });
130 };
131
132 cancelEditHandler = () => {
133     this.setState({ isEditing: false, editPost: null });
134 };
135
136 finishEditHandler = postData => {
137     this.setState({
138         editLoading: true
139     });
140     const formData = new FormData();
141     formData.append('title', postData.title);
142     formData.append('content', postData.content);
143     formData.append('image', postData.image);
144     let url = 'http://localhost:8080/feed/post';
145     let method = 'POST';
146     if (this.state.editPost) {
147         url = 'http://localhost:8080/feed/post/' + this.state.editPost._id;
148         method = 'PUT';
149     }
150
151     fetch(url, {
152         method: method,
153         body: formData,
154         headers: {
155             Authorization: 'Bearer ' + this.props.token
156         }
157     })
158     .then(res => {
159         if (res.status !== 200 && res.status !== 201) {
160             throw new Error('Creating or editing a post failed!');
161         }
162         return res.json();
163     })
164     .then(resData => {
165         console.log(resData);
166         const post = {
167             _id: resData.post._id,
168             title: resData.post.title,
169             content: resData.post.content,
170             creator: resData.post.creator,
171             createdAt: resData.post.createdAt
172         };
173         this.setState(prevState => {
174             let updatedPosts = [...prevState.posts];
175             if (prevState.editPost) {
176                 const postIndex = prevState.posts.findIndex(
177                     p => p._id === prevState.editPost._id
178                 );
179                 updatedPosts[postIndex] = post;

```

```

180     } else if (prevState.posts.length < 2) {
181         updatedPosts = prevState.posts.concat(post);
182     }
183     return {
184         posts: updatedPosts,
185         isEditing: false,
186         editPost: null,
187         editLoading: false
188     };
189 });
190 })
191 .catch(err => {
192     console.log(err);
193     this.setState({
194         isEditing: false,
195         editPost: null,
196         editLoading: false,
197         error: err
198     });
199 });
200 };
201
202 statusInputChangeHandler = (input, value) => {
203     this.setState({ status: value });
204 };
205
206 deletePostHandler = postId => {
207     this.setState({ postsLoading: true });
208     fetch('http://localhost:8080/feed/post/' + postId, {
209         method: 'DELETE',
210         headers: {
211             Authorization: 'Bearer ' + this.props.token
212         }
213     })
214     .then(res => {
215         if (res.status !== 200 && res.status !== 201) {
216             throw new Error('Deleting a post failed!');
217         }
218         return res.json();
219     })
220     .then(resData => {
221         console.log(resData);
222         this.setState(prevState => {
223             const updatedPosts = prevState.posts.filter(p => p._id !== postId);
224             return { posts: updatedPosts, postsLoading: false };
225         });
226     })
227     .catch(err => {
228         console.log(err);
229         this.setState({ postsLoading: false });
230     });
231 };
232
233 errorHandler = () => {
234     this.setState({ error: null });
235 };

```

```

236
237 catchError = error => {
238   this.setState({ error: error });
239 };
240
241 render() {
242   return (
243     <Fragment>
244       <ErrorHandler error={this.state.error} onHandle={this.errorHandler} />
245       <FeedEdit
246         editing={this.state.isEditing}
247         selectedPost={this.state.editPost}
248         loading={this.state.editLoading}
249         onCancelEdit={this.cancelEditHandler}
250         onFinishEdit={this.finishEditHandler}
251       />
252       <section className="feed__status">
253         <form onSubmit={this.statusUpdateHandler}>
254           <Input
255             type="text"
256             placeholder="Your status"
257             control="input"
258             onChange={this.statusInputChangeHandler}
259             value={this.state.status}
260           />
261           <Button mode="flat" type="submit">
262             Update
263           </Button>
264         </form>
265       </section>
266       <section className="feed__control">
267         <Button mode="raised" design="accent" onClick={this.newPostHandler}>
268           New Post
269         </Button>
270       </section>
271       <section className="feed">
272         {this.state.postsLoading && (
273           <div style={{ textAlign: 'center', marginTop: '2rem' }}>
274             <Loader />
275           </div>
276         )}
277         {this.state.posts.length <= 0 && !this.state.postsLoading ? (
278           <p style={{ textAlign: 'center' }}>No posts found.</p>
279         ) : null}
280         {!this.state.postsLoading && (
281           <Paginator
282             onPrevious={this.loadPosts.bind(this, 'previous')}
283             onNext={this.loadPosts.bind(this, 'next')}
284             lastPage={Math.ceil(this.state.totalPosts / 2)}
285             currentPage={this.state.postPage}
286           >
287             {this.state.posts.map(post => (
288               <Post
289                 key={post._id}
290                 id={post._id}
291                 author={post.creator}

```

```

292         date={new Date(post.createdAt).toLocaleDateString('en-US')}
293         title={post.title}
294         image={post.imageUrl}
295         content={post.content}
296         onStartEdit={this.startEditPostHandler.bind(this, post._id)}
297         onDelete={this.deletePostHandler.bind(this, post._id)}
298     />
299     )}
300 </Paginator>
301 )}
302 </section>
303 </Fragment>
304 );
305 }
306 }
307
308 export default Feed;
309

```

```

1  //./controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const Post = require('../models/post');
9  const User = require('../models/user');
10
11 exports.getPosts = async (req, res, next) => {
12     const currentPage = req.query.page || 1;
13     const perPage = 2;
14     let totalItems;
15     try {
16         const totalItems = await Post.find().countDocuments();
17         const posts = await Post.find()
18             .skip((currentPage - 1) * perPage)
19             .limit(perPage);
20
21         res.status(200).json({
22             message: 'Fetched posts successfully.',
23             posts: posts,
24             totalItems: totalItems
25         });
26     } catch (error) {
27         if (!err.statusCode) {
28             err.statusCode = 500;
29         }
30         next(err);
31     }
32 };
33
34 /**we have to go to the place
35  * that is runs or to the code that runs
36  * when a new post is created
37  *
38  * that would be in ./controllers/feed.js(b) to createPost function.

```

```

39 * there i wanna use socket.io the existing connection we have set up
40 * to inform all connected clients about the new post.
41 * for that we need to share that connection
42 * which we currently set up and app.js(b) file
43 */
44 exports.createPost = (req, res, next) => {
45   const errors = validationResult(req);
46   if (!errors.isEmpty()) {
47     const error = new Error('Validation failed, entered data is incorrect.');
```

error.statusCode = 422;

throw error;

```

50   }
51   if (!req.file) {
52     const error = new Error('No image provided.');
```

error.statusCode = 422;

throw error;

```

55   }
56   const imageUrl = req.file.path;
57   const title = req.body.title;
58   const content = req.body.content;
59   let creator;
60   const post = new Post({
61     title: title,
62     content: content,
63     imageUrl: imageUrl,
64     creator: req.userId
65   });
66   post
67     .save()
68     .then(result => {
69       return User.findById(req.userId);
70     })
71     .then(user => {
72       creator = user;
73       user.posts.push(post);
74       return user.save();
75     })
76     .then(result => {
77       res.status(201).json({
78         message: 'Post created successfully!',
79         post: post,
80         creator: { _id: creator._id, name: creator.name }
81       });
82     })
83     .catch(err => {
84       if (!err.statusCode) {
85         err.statusCode = 500;
86       }
87       next(err);
88     });
89   });
90
91 exports.getPost = (req, res, next) => {
92   const postId = req.params.postId;
93   Post.findById(postId)
94     .then(post => {

```

```

95     if (!post) {
96         const error = new Error('Could not find post.');
```

100

```

97         error.statusCode = 404;
98         throw error;
99     }
100     res.status(200).json({ message: 'Post fetched.', post: post });
101 }
102 .catch(err => {
103     if (!err.statusCode) {
104         err.statusCode = 500;
105     }
106     next(err);
107 });
108 };
109
110 exports.updatePost = (req, res, next) => {
111     const postId = req.params.postId;
112     const errors = validationResult(req);
113     if (!errors.isEmpty()) {
114         const error = new Error('Validation failed, entered data is incorrect.');
```

115

```

115         error.statusCode = 422;
116         throw error;
117     }
118     const title = req.body.title;
119     const content = req.body.content;
120     let imageUrl = req.body.image;
121     if (req.file) {
122         imageUrl = req.file.path;
123     }
124     if (!imageUrl) {
125         const error = new Error('No file picked.');
```

126

```

126         error.statusCode = 422;
127         throw error;
128     }
129     Post.findById(postId)
130     .then(post => {
131         if (!post) {
132             const error = new Error('Could not find post.');
```

133

```

133             error.statusCode = 404;
134             throw error;
135         }
136         if (post.creator.toString() !== req.userId) {
137             const error = new Error('Not authorized!');
```

138

```

138             error.statusCode = 403;
139             throw error;
140         }
141         if (imageUrl !== post.imageUrl) {
142             clearImage(post.imageUrl);
143         }
144         post.title = title;
145         post.imageUrl = imageUrl;
146         post.content = content;
147         return post.save();
148     })
149     .then(result => {
150         res.status(200).json({ message: 'Post updated!', post: result });

```

```

151     })
152     .catch(err => {
153       if (!err.statusCode) {
154         err.statusCode = 500;
155       }
156       next(err);
157     });
158   };
159
160   exports.deletePost = (req, res, next) => {
161     const postId = req.params.postId;
162     Post.findById(postId)
163       .then(post => {
164         if (!post) {
165           const error = new Error('Could not find post. ');
166           error.statusCode = 404;
167           throw error;
168         }
169         if (post.creator.toString() !== req.userId) {
170           const error = new Error('Not authorized!');
171           error.statusCode = 403;
172           throw error;
173         }
174         // Check logged in user
175         clearImage(post.imageUrl);
176         return Post.findByIdAndRemove(postId);
177       })
178       .then(result => {
179         return User.findById(req.userId);
180       })
181       .then(user => {
182         user.posts.pull(postId);
183         return user.save();
184       })
185       .then(result => {
186         res.status(200).json({ message: 'Deleted post.' });
187       })
188       .catch(err => {
189         if (!err.statusCode) {
190           err.statusCode = 500;
191         }
192         next(err);
193       });
194   };
195
196   const clearImage = filePath => {
197     filePath = path.join(__dirname, '..', filePath);
198     fs.unlink(filePath, err => console.log(err));
199   };
200

```

## \* Chapter 406: Sharing The IO Instance Across Files

1. update



- socket.js(b)
- app.js(b)

```
1 //socket.js(b)
2
3 let io;
4
5 module.exports = {
6   /**the 'init' method
7    * this is how you define a function in nodejs syntax
8    * you have that key here like in normal object
9    * then a colon and then the value
10   * and the value is a function
11   * which receives the HTTP server as an argument
12   */
13   init: httpServer => {
14     /**like in app.js(b) file
15      * const io = require('socket.io')(server)
16      */
17     io = require('socket.io')(httpServer)
18     return io;
19   },
20   getIO: () => {
21     /**with this, we are managing a connection in this file
22      * and we can import this file in all the places of our app
23      * where we need to be able to interact with IO
24      * like our ./controllers/feed.js(b) file
25      */
26     if (!io) {
27       throw new Error('Socket.io not initialized!');
28     }
29     return io;
30   }
31 }
```

```
1 //app.js(b)
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const multer = require('multer');
9
10 const feedRoutes = require('./routes/feed');
11 const authRoutes = require('./routes/auth');
12
13 const app = express();
14
15 const fileStorage = multer.diskStorage({
16   destination: (req, file, cb) => {
17     cb(null, 'images');
18   },
19   filename: (req, file, cb) => {
20     cb(null, new Date().toISOString() + '-' + file.originalname);
21   }
22 });
```

```

23
24 const fileFilter = (req, file, cb) => {
25   if(
26     file.mimetype === 'image/png' ||
27     file.mimetype === 'image/jpg' ||
28     file.mimetype === 'image/jpeg'
29   ) {
30     cb(null, true);
31   } else {
32     cb(null, false);
33   }
34 }
35
36 // app.use(bodyParser.urlencoded()); // x-www-form-urlencoded <form>
37 app.use(bodyParser.json()); // application/json
38 app.use(multer({storage: fileStorage, fileFilter: fileFilter}).single('image'))
39 app.use('/images', express.static(path.join(__dirname, 'images')));
40
41 app.use((req, res, next) => {
42   res.setHeader('Access-Control-Allow-Origin', '*');
43   res.setHeader(
44     'Access-Control-Allow-Methods',
45     'OPTIONS, GET, POST, PUT, PATCH, DELETE'
46   );
47   res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
48   next();
49 });
50
51 app.use('/feed', feedRoutes);
52 app.use('/auth', authRoutes);
53
54 app.use((error, req, res, next) => {
55   console.log(error);
56   const status = error.statusCode || 500;
57   const message = error.message;
58   const data = error.data;
59   res.status(status).json({ message: message, data: data });
60 });
61
62 mongoose
63   .connect(
64     'mongodb+srv://maximilian:rldnjs12@cluster0-z3vbk.mongodb.net/message?retryWrites=true'
65   )
66   .then(result => {
67     const server = app.listen(8080);
68     /**'init' from socket.js(b) file
69     * because i expect to get that server in init function
70     */
71     const io = require('./socket').init(server);
72     io.on('connection', socket => {
73       console.log('Client connected')
74     })
75   })
76   .catch(err => console.log(err));

```

## \* Chapter 407: Synchronizing POST Additions

1. update
- socket.js(b)
  - app.js(b)

```
1 //socket.js(b)
2
3 let io;
4
5 module.exports = {
6   init: httpServer => {
7     io = require('socket.io')(httpServer)
8     return io;
9   },
10  getIO: () => {
11    if (!io) {
12      throw new Error('Socket.io not initialized!');
13    }
14    return io;
15  }
16 }
```

```
1 //app.js(b)
2
3 const path = require('path');
4
5 const express = require('express');
6 const bodyParser = require('body-parser');
7 const mongoose = require('mongoose');
8 const multer = require('multer');
9
10 const feedRoutes = require('./routes/feed');
11 const authRoutes = require('./routes/auth');
12
13 const app = express();
14
15 const fileStorage = multer.diskStorage({
16   destination: (req, file, cb) => {
17     cb(null, 'images');
18   },
19   filename: (req, file, cb) => {
20     cb(null, new Date().toISOString() + '-' + file.originalname);
21   }
22 });
23
24 const fileFilter = (req, file, cb) => {
25   if(
26     file.mimetype === 'image/png' ||
27     file.mimetype === 'image/jpg' ||
28     file.mimetype === 'image/jpeg'
29   ) {
30     cb(null, true);
31   } else {
32     cb(null, false);
33   }
34 }
```

```

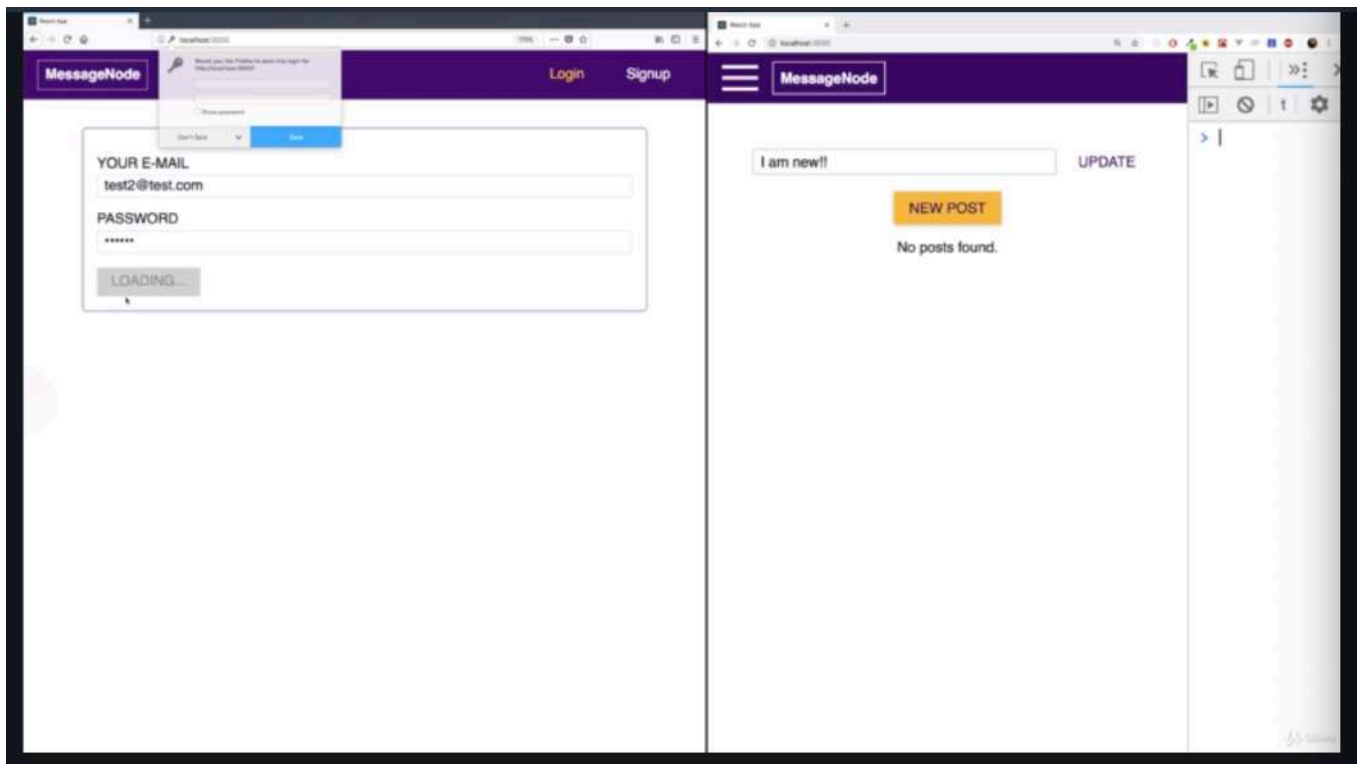
34 }
35
36 // app.use(bodyParser.urlencoded()); // x-www-form-urlencoded <form>
37 app.use(bodyParser.json()); // application/json
38 app.use(multer({storage: fileStorage, fileFilter: fileFilter}).single('image'))
39 app.use('/images', express.static(path.join(__dirname, 'images')));
40
41 app.use((req, res, next) => {
42   res.setHeader('Access-Control-Allow-Origin', '*');
43   res.setHeader(
44     'Access-Control-Allow-Methods',
45     'OPTIONS, GET, POST, PUT, PATCH, DELETE'
46   );
47   res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
48   next();
49 });
50
51 app.use('/feed', feedRoutes);
52 app.use('/auth', authRoutes);
53
54 app.use((error, req, res, next) => {
55   console.log(error);
56   const status = error.statusCode || 500;
57   const message = error.message;
58   const data = error.data;
59   res.status(status).json({ message: message, data: data });
60 });
61
62 mongoose
63   .connect(
64     'mongodb+srv://maximilian:rldnjs12@cluster0-z3v1k.mongodb.net/message?retryWrites=true'
65   )
66   .then(result => {
67     const server = app.listen(8080);
68     const io = require('./socket').init(server);
69     io.on('connection', socket => {
70       console.log('Client connected')
71     })
72   })
73   .catch(err => console.log(err));

```

## \* Chapter 408: Synchronizing POST Additions

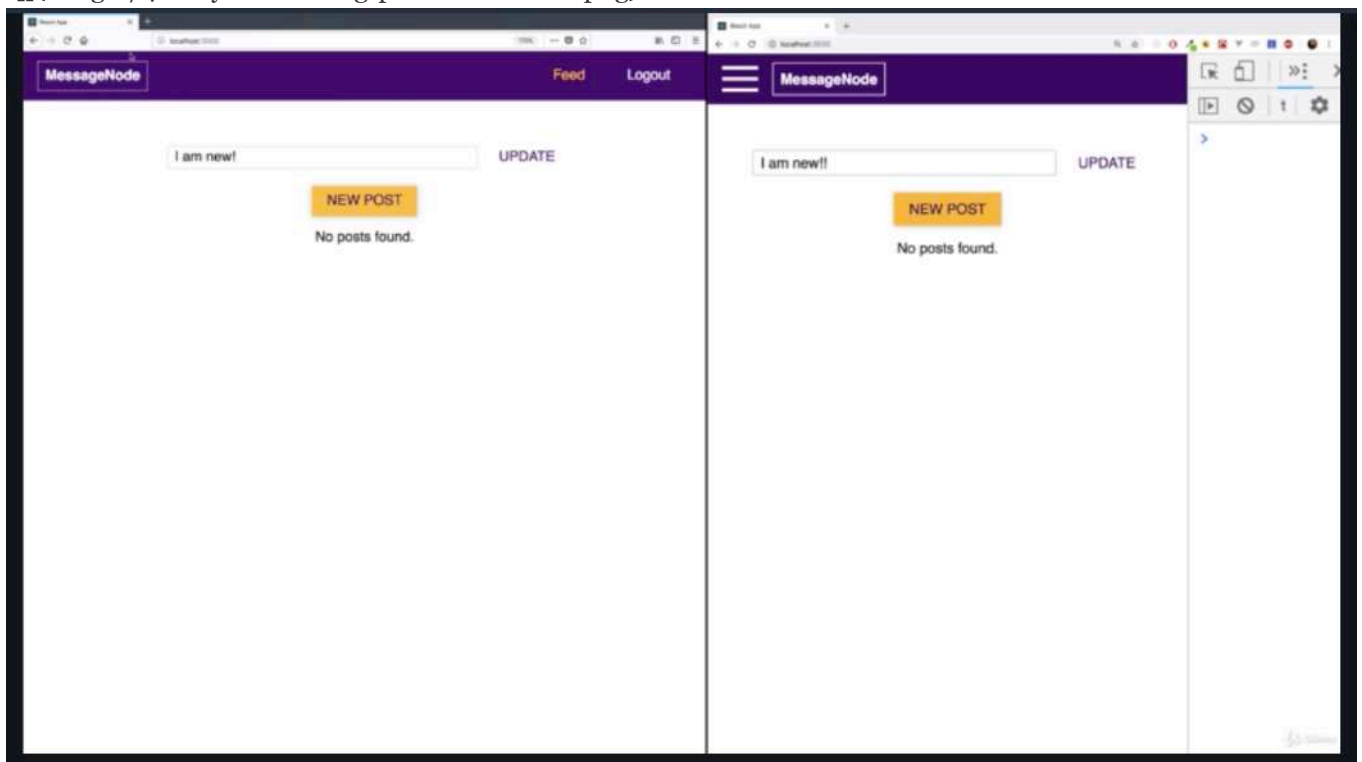
1. update
  - ./controllers/feed.js(b)
  - ./src/pages/Feed/Feed.js(f)





- now to see that in action, i will fake to have a second client so i will open a different browser Firefox and this is the same as if it would on a different PC now.

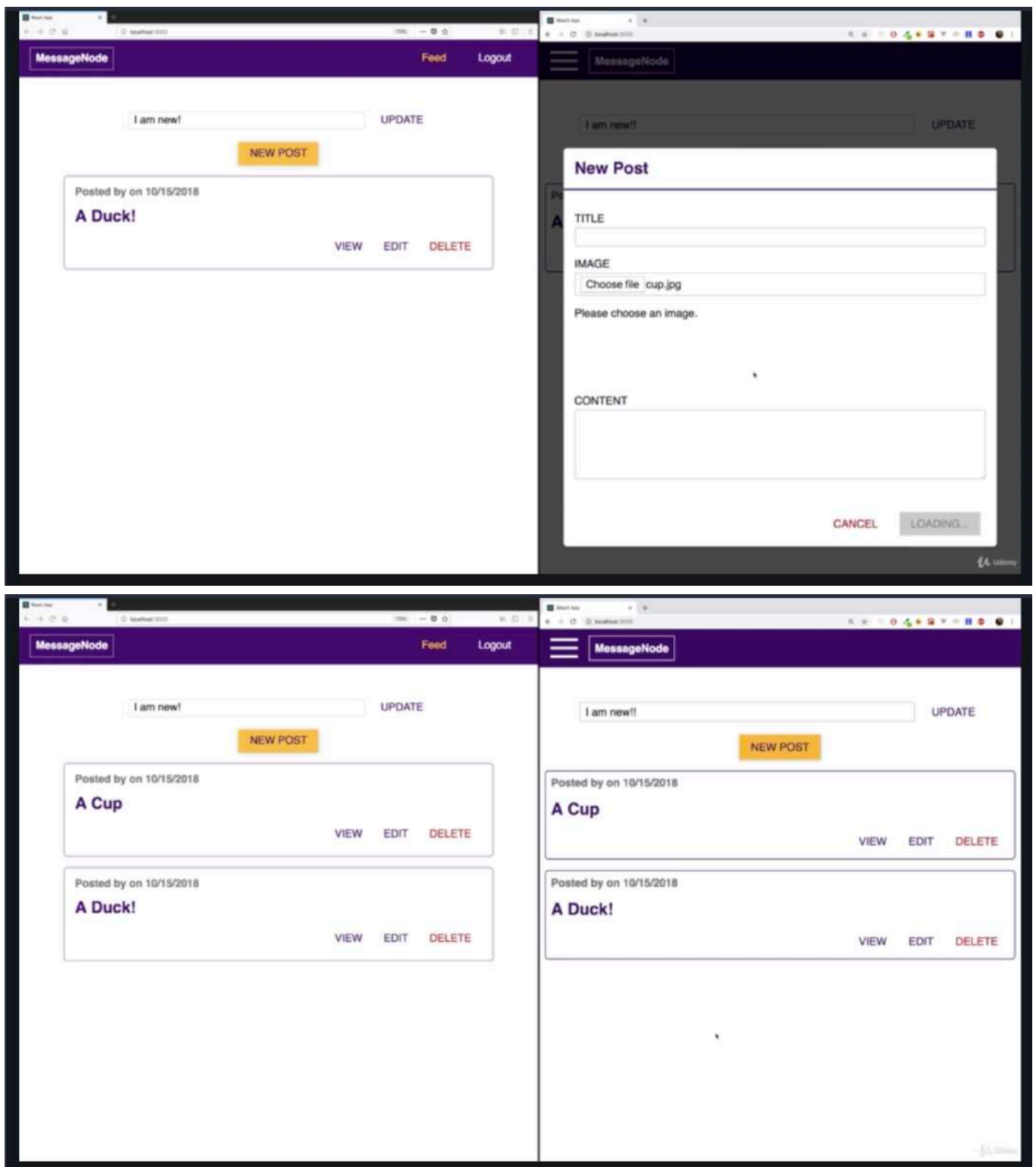




- now i'm logged in on Firefox on the left and on chrome on the right side is 'test@test.com' and user on the left is 'test2@test.com'







- you see it to show up on the left to you also see show up on the right
- i never reloaded the page on left and still we see the new post show up there. that is due to [socket.io](https://socket.io) where we have an established connection on both clients now and thereafter.

```

1  ../controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const io = require('../socket');
9  const Post = require('../models/post');
10 const User = require('../models/user');

```

```

11
12 exports.getPosts = async (req, res, next) => {
13   const currentPage = req.query.page || 1;
14   const perPage = 2;
15   try {
16     const totalItems = await Post.find().countDocuments();
17     const posts = await Post.find()
18       .populate('creator')
19       .skip((currentPage - 1) * perPage)
20       .limit(perPage);
21
22     res.status(200).json({
23       message: 'Fetched posts successfully.',
24       posts: posts,
25       totalItems: totalItems
26     });
27   } catch (err) {
28     if (!err.statusCode) {
29       err.statusCode = 500;
30     }
31     next(err);
32   }
33 };
34
35 exports.createPost = async (req, res, next) => {
36   const errors = validationResult(req);
37   if (!errors.isEmpty()) {
38     const error = new Error('Validation failed, entered data is incorrect.');
```

error.statusCode = 422;

```

40     throw error;
41   }
42   if (!req.file) {
43     const error = new Error('No image provided.');
```

error.statusCode = 422;

```

45     throw error;
46   }
47   const imageUrl = req.file.path;
48   const title = req.body.title;
49   const content = req.body.content;
50   const post = new Post({
51     title: title,
52     content: content,
53     imageUrl: imageUrl,
54     creator: req.userId
55   });
56   try {
57     await post.save();
58     const user = await User.findById(req.userId);
59     user.posts.push(post);
60     await user.save();
61     /**'emit()' are all would be broadcast the differences
62     * that emit will now send a message to all connected users.
63     */
64     io.getIO().emit('posts', {
65       /**i will define action key which inform the client
66       * what happened to the channel is posts

```



```

67      * but action is 'create'
68      * that is one way of implementing this.
69      *
70      * i'm sending this post object in this data package on this 'posts' channel
71      * we are sending this to all connected the clients
72      */
73      action: 'create',
74      post: { ...post._doc, creator: { _id: req.userId, name: user.name } }
75    });
76    res.status(201).json({
77      message: 'Post created successfully!',
78      post: post,
79      creator: { _id: user._id, name: user.name }
80    });
81  } catch (err) {
82    if (!err.statusCode) {
83      err.statusCode = 500;
84    }
85    next(err);
86  }
87 };
88
89 exports.getPost = async (req, res, next) => {
90   const postId = req.params.postId;
91   const post = await Post.findById(postId);
92   try {
93     if (!post) {
94       const error = new Error('Could not find post. ');
95       error.statusCode = 404;
96       throw error;
97     }
98     res.status(200).json({ message: 'Post fetched.', post: post });
99   } catch (err) {
100     if (!err.statusCode) {
101       err.statusCode = 500;
102     }
103     next(err);
104   }
105 };
106
107 exports.updatePost = async (req, res, next) => {
108   const postId = req.params.postId;
109   const errors = validationResult(req);
110   if (!errors.isEmpty()) {
111     const error = new Error('Validation failed, entered data is incorrect. ');
112     error.statusCode = 422;
113     throw error;
114   }
115   const title = req.body.title;
116   const content = req.body.content;
117   let imageUrl = req.body.image;
118   if (req.file) {
119     imageUrl = req.file.path;
120   }
121   if (!imageUrl) {
122     const error = new Error('No file picked. ');

```

```

123     error.statusCode = 422;
124     throw error;
125 }
126 try {
127     const post = await Post.findById(postId);
128     if (!post) {
129         const error = new Error('Could not find post.');
```

130 error.statusCode = 404;

131 throw error;

132 }

133 if (post.creator.toString() !== req.userId) {

134 const error = new Error('Not authorized!');

135 error.statusCode = 403;

136 throw error;

137 }

138 if (imageUrl !== post.imageUrl) {

139 clearImage(post.imageUrl);

140 }

141 post.title = title;

142 post.imageUrl = imageUrl;

143 post.content = content;

144 const result = await post.save();

145 res.status(200).json({ message: 'Post updated!', post: result });

146 } catch (err) {

147 if (!err.statusCode) {

148 err.statusCode = 500;

149 }

150 next(err);

151 }

152 };

153

154 exports.deletePost = async (req, res, next) => {

155 const postId = req.params.postId;

156 try {

157 const post = await Post.findById(postId);

158

159 if (!post) {

160 const error = new Error('Could not find post.');

161 error.statusCode = 404;

162 throw error;

163 }

164 if (post.creator.toString() !== req.userId) {

165 const error = new Error('Not authorized!');

166 error.statusCode = 403;

167 throw error;

168 }

169 // Check logged in user

170 clearImage(post.imageUrl);

171 await Post.findByIdAndRemove(postId);

172

173 const user = await User.findById(req.userId);

174 user.posts.pull(postId);

175 await user.save();

176

177 res.status(200).json({ message: 'Deleted post.' });

178 } catch (err) {

```

179     if (!err.statusCode) {
180       err.statusCode = 500;
181     }
182     next(err);
183   }
184 };
185
186 const clearImage = filePath => {
187   filePath = path.join(__dirname, '..', filePath);
188   fs.unlink(filePath, err => console.log(err));
189 };
190

```

```

1  //./src/pages/Feed/Feed.js(f)
2
3  import React, { Component, Fragment } from 'react';
4  import openSocket from 'socket.io-client';
5
6  import Post from '../../components/Feed/Post/Post';
7  import Button from '../../components/Button/Button';
8  import FeedEdit from '../../components/Feed/FeedEdit/FeedEdit';
9  import Input from '../../components/Form/Input/Input';
10 import Paginator from '../../components/Paginator/Paginator';
11 import Loader from '../../components/Loader/Loader';
12 import ErrorHandler from '../../components/ErrorHandler/ErrorHandler';
13 import './Feed.css';
14
15 class Feed extends Component {
16   state = {
17     isEditing: false,
18     posts: [],
19     totalPosts: 0,
20     editPost: null,
21     status: '',
22     postPage: 1,
23     postsLoading: true,
24     editLoading: false
25   };
26
27   componentDidMount() {
28     fetch('http://localhost:8080/auth/status', {
29       headers: {
30         Authorization: 'Bearer ' + this.props.token
31       }
32     })
33       .then(res => {
34         if (res.status !== 200) {
35           throw new Error('Failed to fetch user status.');

```

```

45     /**in componentDidMount,
46     * after opening my socket
47     * i would store something which is returned by opensocket
48     * so the socket the connection which was opened.
49     *
50     * and on that socket,
51     * we can use to 'on()' method to listen to certain events.
52     * now we wanna use that same event name we used on the backend
53     */
54     const socket = openSocket('http://localhost:8080');
55     socket.on('posts', data => {
56         /**in here, i know due to setup,
57         * i chose my object which i send back
58         * and will have the action key that define what happened to posts
59         * but that's just the pattern i established
60         * that is not enforced by socket.io
61         * you can send any data you want there
62         * you don't have a action key
63         *
64         * but i want it
65         * because i will send different events
66         * or different kinds of operations on posts
67         * through the same channel eventually.
68         */
69         if (data.action === 'create') {
70             this.addPost(data.post);
71         }
72     });
73 }
74
75 addPost = post => {
76     this.setState(prevState => {
77         const updatedPosts = [...prevState.posts];
78         if (prevState.postPage === 1) {
79             if (prevState.posts.length >= 2) {
80                 updatedPosts.pop();
81             }
82             updatedPosts.unshift(post);
83         }
84         return {
85             posts: updatedPosts,
86             totalPosts: prevState.totalPosts + 1
87         };
88     });
89 };
90
91 loadPosts = direction => {
92     if (direction) {
93         this.setState({ postsLoading: true, posts: [] });
94     }
95     let page = this.state.postPage;
96     if (direction === 'next') {
97         page++;
98         this.setState({ postPage: page });
99     }
100     if (direction === 'previous') {

```

```

101     page--;
102     this.setState({ postPage: page });
103 }
104 fetch('http://localhost:8080/feed/posts?page=' + page, {
105     headers: {
106         Authorization: 'Bearer ' + this.props.token
107     }
108 })
109     .then(res => {
110         if (res.status !== 200) {
111             throw new Error('Failed to fetch posts.');

```

```

157
158 startEditPostHandler = postId => {
159   this.setState(prevState => {
160     const loadedPost = { ...prevState.posts.find(p => p._id === postId) };
161
162     return {
163       isEditing: true,
164       editPost: loadedPost
165     };
166   });
167 };
168
169 cancelEditHandler = () => {
170   this.setState({ isEditing: false, editPost: null });
171 };
172
173 finishEditHandler = postData => {
174   this.setState({
175     editLoading: true
176   });
177   const formData = new FormData();
178   formData.append('title', postData.title);
179   formData.append('content', postData.content);
180   formData.append('image', postData.image);
181   let url = 'http://localhost:8080/feed/post';
182   let method = 'POST';
183   if (this.state.editPost) {
184     url = 'http://localhost:8080/feed/post/' + this.state.editPost._id;
185     method = 'PUT';
186   }
187
188   fetch(url, {
189     method: method,
190     body: formData,
191     headers: {
192       Authorization: 'Bearer ' + this.props.token
193     }
194   })
195     .then(res => {
196       if (res.status !== 200 && res.status !== 201) {
197         throw new Error('Creating or editing a post failed!');
198       }
199       return res.json();
200     })
201     .then(resData => {
202       console.log(resData);
203       const post = {
204         _id: resData.post._id,
205         title: resData.post.title,
206         content: resData.post.content,
207         creator: resData.post.creator,
208         createdAt: resData.post.createdAt
209       };
210       this.setState(prevState => {
211         let updatedPosts = [...prevState.posts];
212         if (prevState.editPost) {

```

```

213         const postIndex = prevState.posts.findIndex(
214             p => p._id === prevState.editPost._id
215         );
216         updatedPosts[postIndex] = post;
217     }
218     return {
219         posts: updatedPosts,
220         isEditing: false,
221         editPost: null,
222         editLoading: false
223     };
224 });
225 })
226 .catch(err => {
227     console.log(err);
228     this.setState({
229         isEditing: false,
230         editPost: null,
231         editLoading: false,
232         error: err
233     });
234 });
235 };
236
237 statusInputChangeHandler = (input, value) => {
238     this.setState({ status: value });
239 };
240
241 deletePostHandler = postId => {
242     this.setState({ postsLoading: true });
243     fetch('http://localhost:8080/feed/post/' + postId, {
244         method: 'DELETE',
245         headers: {
246             Authorization: 'Bearer ' + this.props.token
247         }
248     })
249     .then(res => {
250         if (res.status !== 200 && res.status !== 201) {
251             throw new Error('Deleting a post failed!');
252         }
253         return res.json();
254     })
255     .then(resData => {
256         console.log(resData);
257         this.setState(prevState => {
258             const updatedPosts = prevState.posts.filter(p => p._id !== postId);
259             return { posts: updatedPosts, postsLoading: false };
260         });
261     })
262     .catch(err => {
263         console.log(err);
264         this.setState({ postsLoading: false });
265     });
266 };
267
268 errorHandler = () => {

```



```

269     this.setState({ error: null });
270 };
271
272 catchError = error => {
273     this.setState({ error: error });
274 };
275
276 render() {
277     return (
278         <Fragment>
279             <ErrorHandler error={this.state.error} onHandle={this.errorHandler} />
280             <FeedEdit
281                 editing={this.state.isEditing}
282                 selectedPost={this.state.editPost}
283                 loading={this.state.editLoading}
284                 onCancelEdit={this.cancelEditHandler}
285                 onFinishEdit={this.finishEditHandler}
286             />
287             <section className="feed__status">
288                 <form onSubmit={this.statusUpdateHandler}>
289                     <Input
290                         type="text"
291                         placeholder="Your status"
292                         control="input"
293                         onChange={this.statusInputChangeHandler}
294                         value={this.state.status}
295                     />
296                     <Button mode="flat" type="submit">
297                         Update
298                     </Button>
299                 </form>
300             </section>
301             <section className="feed__control">
302                 <Button mode="raised" design="accent" onClick={this.newPostHandler}>
303                     New Post
304                 </Button>
305             </section>
306             <section className="feed">
307                 {this.state.postsLoading && (
308                     <div style={{ textAlign: 'center', marginTop: '2rem' }}>
309                         <Loader />
310                     </div>
311                 )}
312                 {this.state.posts.length <= 0 && !this.state.postsLoading ? (
313                     <p style={{ textAlign: 'center' }}>No posts found.</p>
314                 ) : null}
315                 {!this.state.postsLoading && (
316                     <Paginator
317                         onPrevious={this.loadPosts.bind(this, 'previous')}
318                         onNext={this.loadPosts.bind(this, 'next')}
319                         lastPage={Math.ceil(this.state.totalPosts / 2)}
320                         currentPage={this.state.postPage}
321                     >
322                         {this.state.posts.map(post => (
323                             <Post
324                                 key={post._id}

```

```

325     id={post._id}
326     author={post.creator.name}
327     date={new Date(post.createdAt).toLocaleDateString('en-US')}
328     title={post.title}
329     image={post.imageUrl}
330     content={post.content}
331     onStartEdit={this.startEditPostHandler.bind(this, post._id)}
332     onDelete={this.deletePostHandler.bind(this, post._id)}
333   />
334   ))}
335 </Paginator>
336 )}
337 </section>
338 </Fragment>
339 );
340 }
341 }
342
343 export default Feed;
344

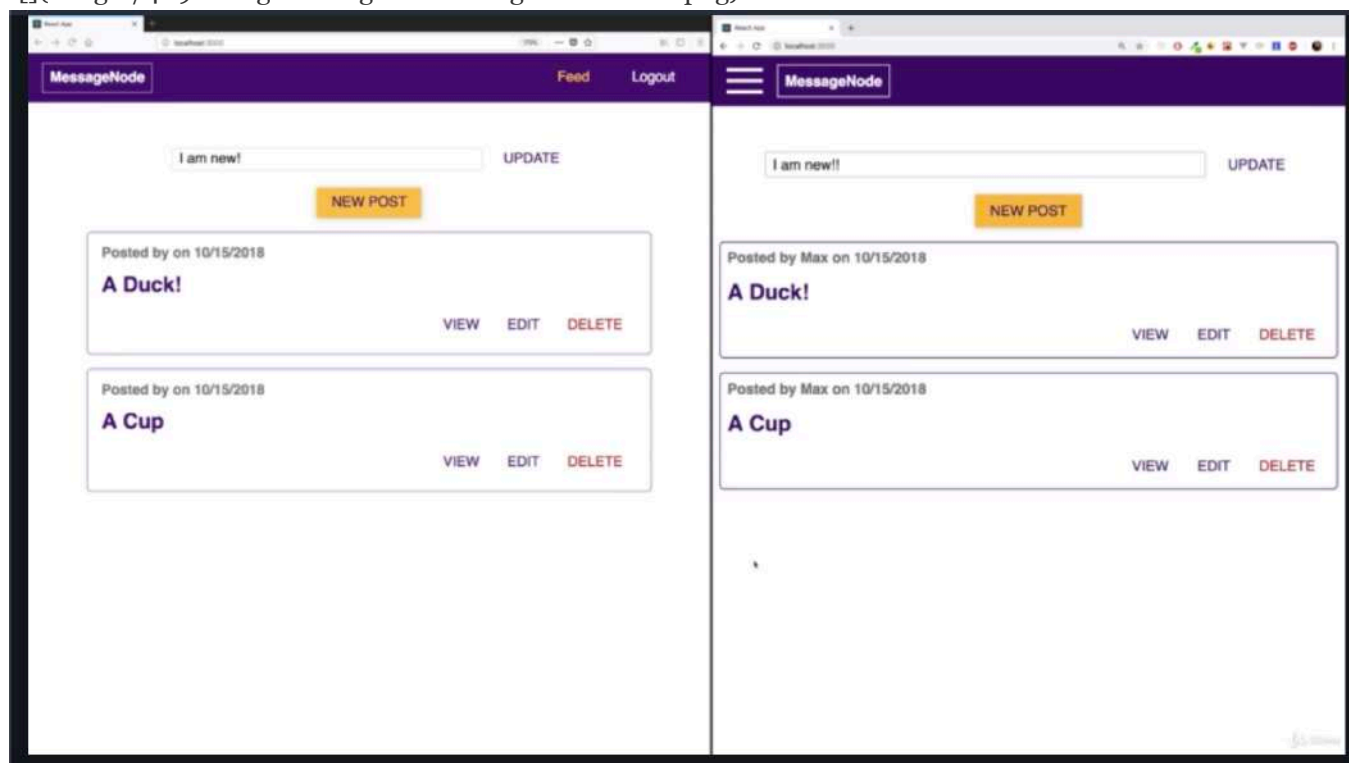
```

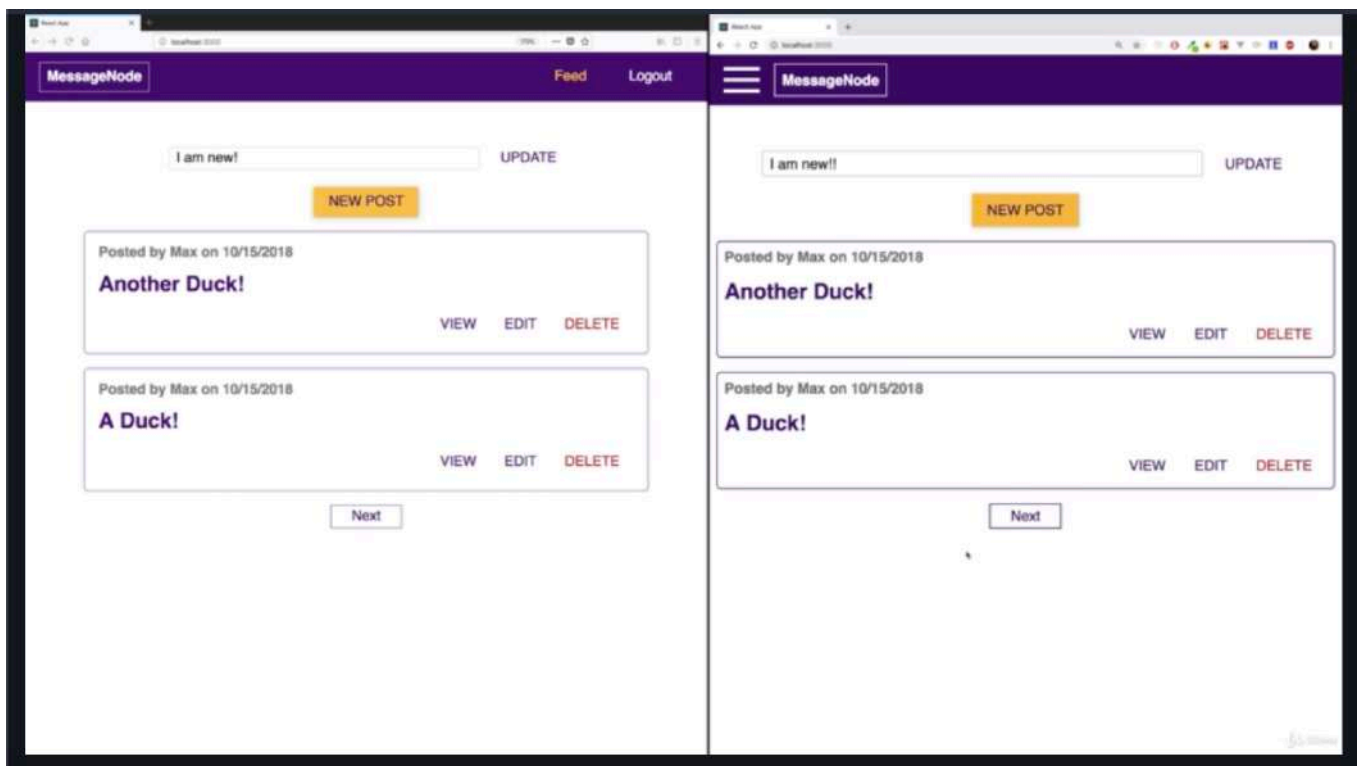
## \* Chapter 409: Fixing The Bug - The Missing Username

1. update  
- ./controllers/feed.js(b)









```

1  ../controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const io = require('../socket');
9  const Post = require('../models/post');
10 const User = require('../models/user');
11
12 exports.getPosts = async (req, res, next) => {
13   const currentPage = req.query.page || 1;
14   const perPage = 2;
15   try {
16     const totalItems = await Post.find().countDocuments();
17     const posts = await Post.find()
18       .populate('creator')
19       .skip((currentPage - 1) * perPage)
20       .limit(perPage);
21
22     res.status(200).json({
23       message: 'Fetched posts successfully.',
24       posts: posts,
25       totalItems: totalItems
26     });
27   } catch (err) {
28     if (!err.statusCode) {
29       err.statusCode = 500;
30     }
31     next(err);
32   }
33 };
34
35 exports.createPost = async (req, res, next) => {

```

```

36 const errors = validationResult(req);
37 if (!errors.isEmpty()) {
38   const error = new Error('Validation failed, entered data is incorrect.');
```

39 error.statusCode = 422;
40 throw error;
41 }
42 if (!req.file) {
43 const error = new Error('No image provided.');

44 error.statusCode = 422;
45 throw error;
46 }
47 const imageUrl = req.file.path;
48 const title = req.body.title;
49 const content = req.body.content;
50 const post = new Post({
51 title: title,
52 content: content,
53 imageUrl: imageUrl,
54 creator: req.userId
55 });
56 try {
57 await post.save();
58 const user = await User.findById(req.userId);
59 user.posts.push(post);
60 await user.save();
61 io.getIO().emit('posts', {
62 action: 'create',
63 /\*\*i'm sending socket.io is just a post object
64 \* where i did at my 'creator: req.userId'
65 \* but not all the creator data.
66 \*
67 \* we can send '...post.\_doc' data
68 \* so all the data about the post
69 \*/
70 post: {
71 ...post.\_doc,
72 creator: { \_id: req.userId, name: user.name } }
73 });
74 res.status(201).json({
75 message: 'Post created successfully!',
76 post: post,
77 creator: { \_id: user.\_id, name: user.name }
78 });
79 } catch (err) {
80 if (!err.statusCode) {
81 err.statusCode = 500;
82 }
83 next(err);
84 }
85 };
86
87 exports.getPost = async (req, res, next) => {
88 const postId = req.params.postId;
89 const post = await Post.findById(postId);
90 try {
91 if (!post) {

```

92     const error = new Error('Could not find post.');
```

```

93     error.statusCode = 404;
```

```

94     throw error;
```

```

95 }
96 res.status(200).json({ message: 'Post fetched.', post: post });
97 } catch (err) {
98     if (!err.statusCode) {
99         err.statusCode = 500;
100     }
101     next(err);
102 }
103 };
104
105 exports.updatePost = async (req, res, next) => {
106     const postId = req.params.postId;
107     const errors = validationResult(req);
108     if (!errors.isEmpty()) {
109         const error = new Error('Validation failed, entered data is incorrect.');
```

```

110         error.statusCode = 422;
```

```

111         throw error;
```

```

112     }
113     const title = req.body.title;
```

```

114     const content = req.body.content;
```

```

115     let imageUrl = req.body.image;
```

```

116     if (req.file) {
117         imageUrl = req.file.path;
```

```

118     }
119     if (!imageUrl) {
120         const error = new Error('No file picked.');
```

```

121         error.statusCode = 422;
```

```

122         throw error;
```

```

123     }
124     try {
125         const post = await Post.findById(postId);
126         if (!post) {
127             const error = new Error('Could not find post.');
```

```

128             error.statusCode = 404;
```

```

129             throw error;
```

```

130         }
131         if (post.creator.toString() !== req.userId) {
132             const error = new Error('Not authorized!');
```

```

133             error.statusCode = 403;
```

```

134             throw error;
```

```

135         }
136         if (imageUrl !== post.imageUrl) {
137             clearImage(post.imageUrl);
138         }
139         post.title = title;
```

```

140         post.imageUrl = imageUrl;
```

```

141         post.content = content;
```

```

142         const result = await post.save();
143         res.status(200).json({ message: 'Post updated!', post: result });
144     } catch (err) {
145         if (!err.statusCode) {
146             err.statusCode = 500;
```

```

147         }

```

```

148     next(err);
149   }
150 };
151
152 exports.deletePost = async (req, res, next) => {
153   const postId = req.params.postId;
154   try {
155     const post = await Post.findById(postId);
156
157     if (!post) {
158       const error = new Error('Could not find post.');
```

```

159       error.statusCode = 404;
160       throw error;
161     }
162     if (post.creator.toString() !== req.userId) {
163       const error = new Error('Not authorized!');
```

```

164       error.statusCode = 403;
165       throw error;
166     }
167     // Check logged in user
168     clearImage(post.imageUrl);
169     await Post.findByIdAndRemove(postId);
170
171     const user = await User.findById(req.userId);
172     user.posts.pull(postId);
173     await user.save();
174
175     res.status(200).json({ message: 'Deleted post.' });
176   } catch (err) {
177     if (!err.statusCode) {
178       err.statusCode = 500;
179     }
180     next(err);
181   }
182 };
183
184 const clearImage = filePath => {
185   filePath = path.join(__dirname, '..', filePath);
186   fs.unlink(filePath, err => console.log(err));
187 };
188
```

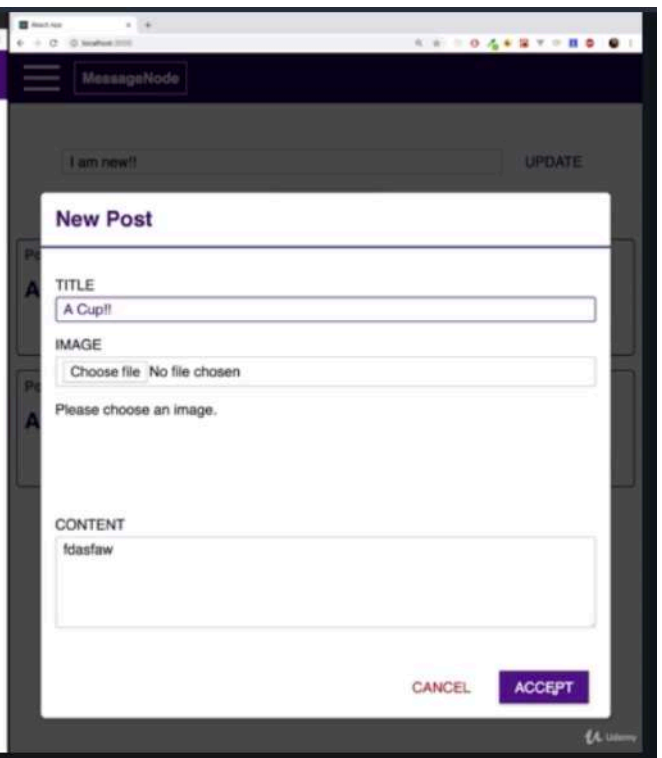
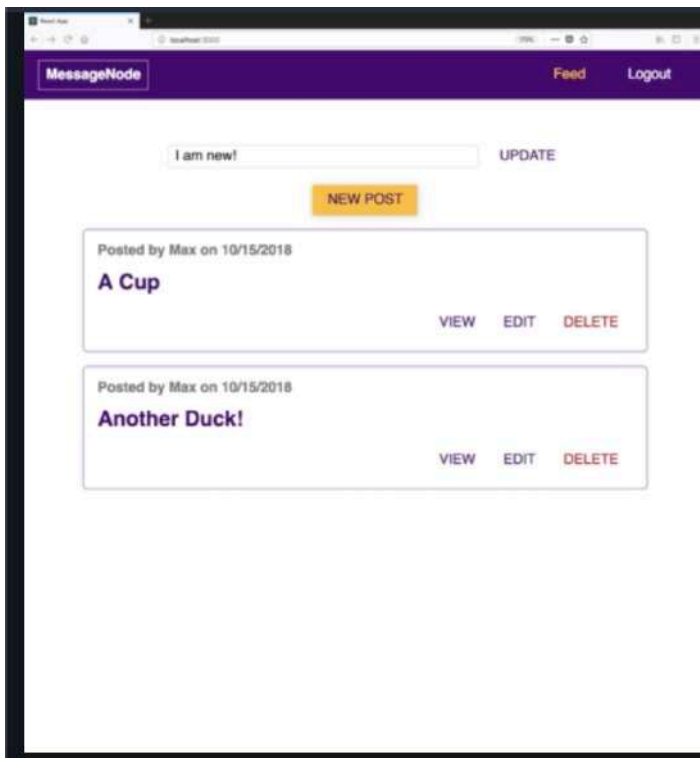
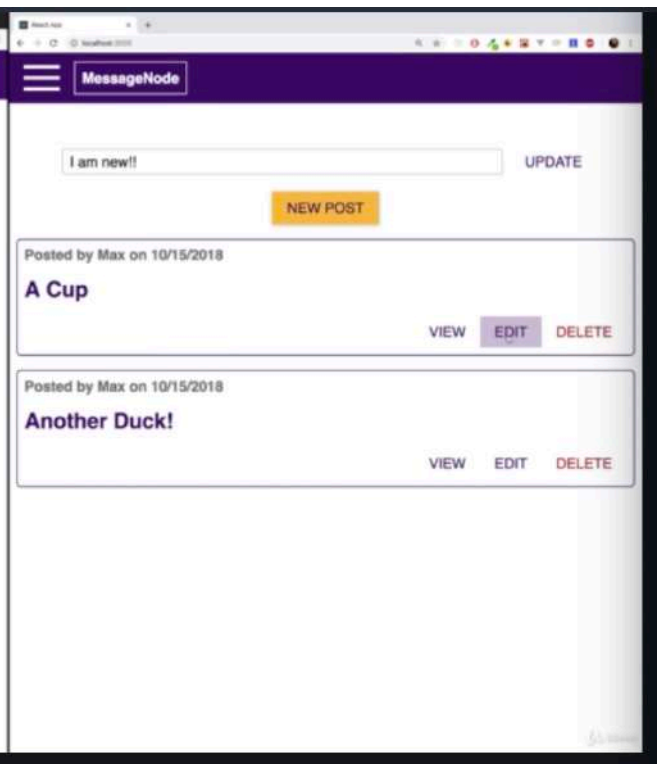
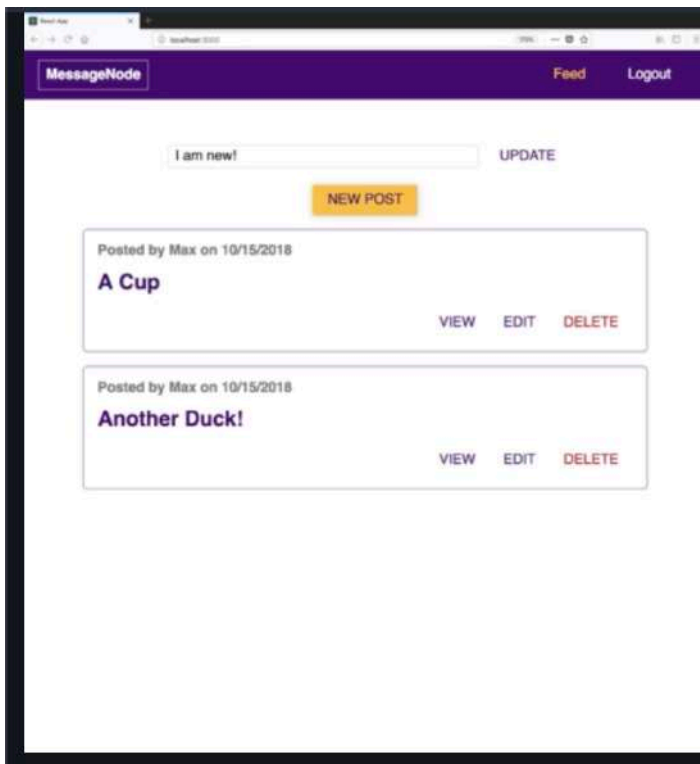
## \* Chapter 410: Updating Posts On All Connected Clients

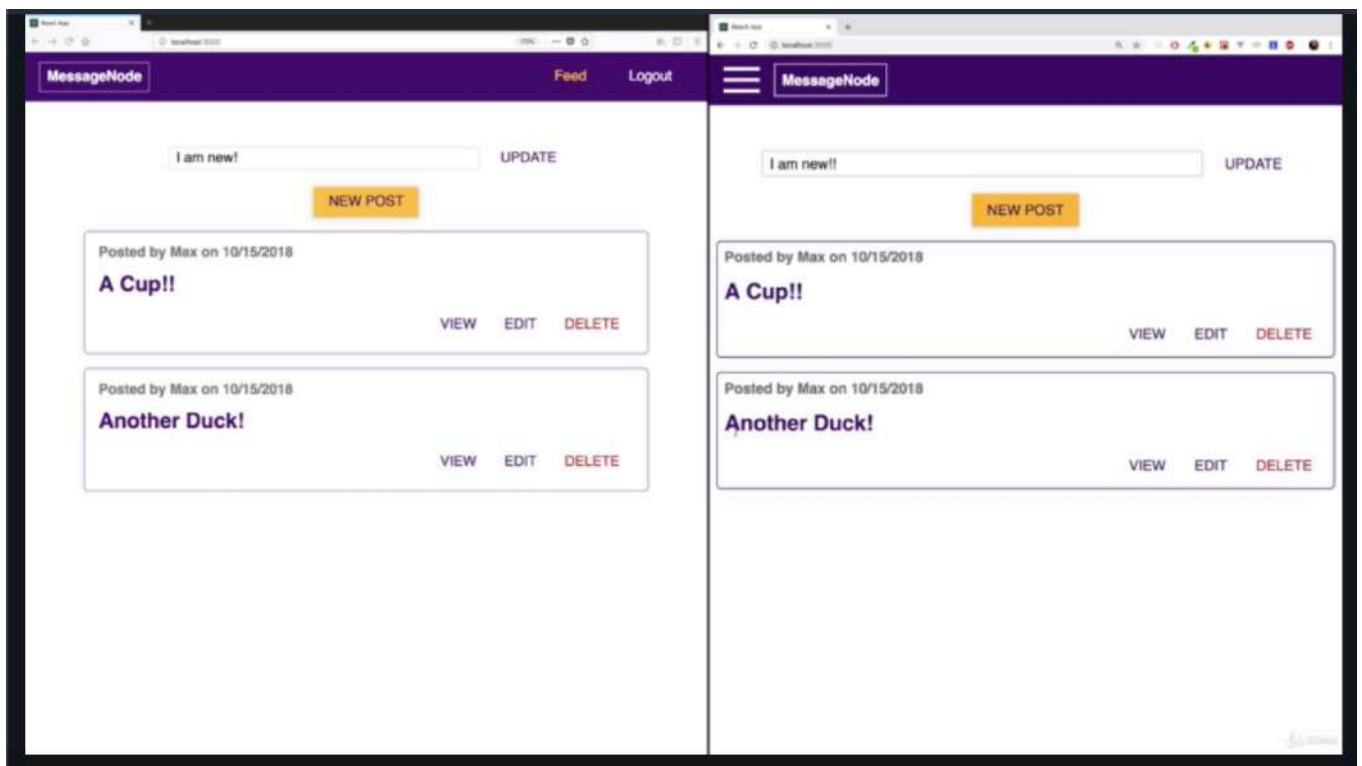
1. update
  - ./controllers/feed.js(b)
  - ./src/pages/Feed/Feed.js(f)











```

1  ../controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const io = require('../socket');
9  const Post = require('../models/post');
10 const User = require('../models/user');
11
12 exports.getPosts = async (req, res, next) => {
13   const currentPage = req.query.page || 1;
14   const perPage = 2;
15   try {
16     const totalItems = await Post.find().countDocuments();
17     const posts = await Post.find()
18       .populate('creator')
19       .skip((currentPage - 1) * perPage)
20       .limit(perPage);
21
22     res.status(200).json({
23       message: 'Fetched posts successfully.',
24       posts: posts,
25       totalItems: totalItems
26     });
27   } catch (err) {
28     if (!err.statusCode) {
29       err.statusCode = 500;
30     }
31     next(err);
32   }
33 };
34
35 exports.createPost = async (req, res, next) => {

```



```

36 const errors = validationResult(req);
37 if (!errors.isEmpty()) {
38   const error = new Error('Validation failed, entered data is incorrect.');
```

```

39   error.statusCode = 422;
40   throw error;
41 }
42 if (!req.file) {
43   const error = new Error('No image provided.');
```

```

44   error.statusCode = 422;
45   throw error;
46 }
47 const imageUrl = req.file.path;
48 const title = req.body.title;
49 const content = req.body.content;
50 const post = new Post({
51   title: title,
52   content: content,
53   imageUrl: imageUrl,
54   creator: req.userId
55 });
56 try {
57   await post.save();
58   const user = await User.findById(req.userId);
59   user.posts.push(post);
60   await user.save();
61   io.getIO().emit('posts', {
62     action: 'create',
63     post: {
64       ...post._doc,
65       creator: { _id: req.userId, name: user.name } }
66   });
67   res.status(201).json({
68     message: 'Post created successfully!',
69     post: post,
70     creator: { _id: user._id, name: user.name }
71   });
72 } catch (err) {
73   if (!err.statusCode) {
74     err.statusCode = 500;
75   }
76   next(err);
77 }
78 };
79
80 exports.getPost = async (req, res, next) => {
81   const postId = req.params.postId;
82   const post = await Post.findById(postId);
83   try {
84     if (!post) {
85       const error = new Error('Could not find post.');
```

```

86       error.statusCode = 404;
87       throw error;
88     }
89     res.status(200).json({ message: 'Post fetched.', post: post });
90   } catch (err) {
91     if (!err.statusCode) {

```

```

92     err.statusCode = 500;
93   }
94   next(err);
95 }
96 };
97
98 exports.updatePost = async (req, res, next) => {
99   const postId = req.params.postId;
100  const errors = validationResult(req);
101  if (!errors.isEmpty()) {
102    const error = new Error('Validation failed, entered data is incorrect.');
```

error.statusCode = 422;

```

104    throw error;
105  }
106  const title = req.body.title;
107  const content = req.body.content;
108  let imageUrl = req.body.image;
109  if (req.file) {
110    imageUrl = req.file.path;
111  }
112  if (!imageUrl) {
113    const error = new Error('No file picked.');
```

error.statusCode = 422;

```

115    throw error;
116  }
117  try {
118    /**'populate('creator')' will take that creator ID
119     * which we stored in the post object
120     * reach out to the users database for the user collection
121     * fetched data for that specific user
122     * and added here in our post
123     */
124    const post = await Post.findById(postId).populate('creator');
```

if (!post) {

```

126    const error = new Error('Could not find post.');
```

error.statusCode = 404;

```

128    throw error;
129  }
130  /**not just 'post.creator.toString()'
131   * but just 'post.creator._id.toString()'
132   * because we are populating creator field with the full user data
133   * so not just id anymore
134   */
135  if (post.creator._id.toString() !== req.userId) {
136    const error = new Error('Not authorized!');
```

error.statusCode = 403;

```

138    throw error;
139  }
140  if (imageUrl !== post.imageUrl) {
141    clearImage(post.imageUrl);
142  }
143  post.title = title;
144  post.imageUrl = imageUrl;
145  post.content = content;
146  const result = await post.save();
147  io.getIO().emit('posts', { action: 'update', post: result });

```

```

148     res.status(200).json({ message: 'Post updated!', post: result });
149   } catch (err) {
150     if (!err.statusCode) {
151       err.statusCode = 500;
152     }
153     next(err);
154   }
155 };
156
157 exports.deletePost = async (req, res, next) => {
158   const postId = req.params.postId;
159   try {
160     const post = await Post.findById(postId);
161
162     if (!post) {
163       const error = new Error('Could not find post.');
```

```

164       error.statusCode = 404;
165       throw error;
166     }
167     if (post.creator.toString() !== req.userId) {
168       const error = new Error('Not authorized!');
```

```

11 import Loader from '../components/Loader/Loader';
12 import ErrorHandler from '../components/ErrorHandler/ErrorHandler';
13 import './Feed.css';
14
15 class Feed extends Component {
16   state = {
17     isEditing: false,
18     posts: [],
19     totalPosts: 0,
20     editPost: null,
21     status: '',
22     postPage: 1,
23     postsLoading: true,
24     editLoading: false
25   };
26
27   componentDidMount() {
28     fetch('http://localhost:8080/auth/status', {
29       headers: {
30         Authorization: 'Bearer ' + this.props.token
31       }
32     })
33       .then(res => {
34         if (res.status !== 200) {
35           throw new Error('Failed to fetch user status.');

```

```

67      * through the same channel eventually.
68      */
69      if (data.action === 'create') {
70          this.addPost(data.post);
71      } else if (data.action === 'update') {
72          this.updatePost(data.post);
73      }
74      });
75  }
76
77  addPost = post => {
78      this.setState(prevState => {
79          const updatedPosts = [...prevState.posts];
80          if (prevState.postPage === 1) {
81              if (prevState.posts.length >= 2) {
82                  updatedPosts.pop();
83              }
84              updatedPosts.unshift(post);
85          }
86          return {
87              posts: updatedPosts,
88              totalPosts: prevState.totalPosts + 1
89          };
90      });
91  };
92
93  updatePost = post => {
94      this.setState(prevState => {
95          const updatedPosts = [...prevState.posts];
96          const updatedPostIndex = updatedPosts.findIndex(p => p._id === post._id);
97          if (updatedPostIndex > -1) {
98              updatedPosts[updatedPostIndex] = post;
99          }
100         return {
101             posts: updatedPosts
102         };
103     });
104 }
105
106 loadPosts = direction => {
107     if (direction) {
108         this.setState({ postsLoading: true, posts: [] });
109     }
110     let page = this.state.postPage;
111     if (direction === 'next') {
112         page++;
113         this.setState({ postPage: page });
114     }
115     if (direction === 'previous') {
116         page--;
117         this.setState({ postPage: page });
118     }
119     fetch('http://localhost:8080/feed/posts?page=' + page, {
120         headers: {
121             Authorization: 'Bearer ' + this.props.token
122         }

```

```

123     })
124     .then(res => {
125       if (res.status !== 200) {
126         throw new Error('Failed to fetch posts.');
```

127 }

128 return res.json();

129 })

130 .then(resData => {

131 this.setState({

132 posts: resData.posts.map(post => {

133 return {

134 ...post,

135 imagePath: post.imageUrl

136 };

137 } ),

138 totalPosts: resData.totalItems,

139 postsLoading: false

140 });

141 })

142 .catch(this.catchError);

143 };
144
145 statusUpdateHandler = event => {
146 event.preventDefault();
147 fetch('http://localhost:8080/auth/status', {
148 method: 'PATCH',
149 headers: {
150 Authorization: 'Bearer ' + this.props.token,
151 'Content-Type': 'application/json'
152 },
153 body: JSON.stringify({
154 status: this.state.status
155 })
156 })
157 .then(res => {
158 if (res.status !== 200 && res.status !== 201) {
159 throw new Error("Can't update status!");
160 }
161 return res.json();
162 })
163 .then(resData => {
164 console.log(resData);
165 })
166 .catch(this.catchError);
167 };
168
169 newPostHandler = () => {
170 this.setState({ isEditing: true });
171 };
172
173 startEditPostHandler = postId => {
174 this.setState(prevState => {
175 const loadedPost = { ...prevState.posts.find(p => p.\_id === postId) };
176
177 return {
178 isEditing: true,

```

179         editPost: loadedPost
180     };
181 });
182 };
183
184 cancelEditHandler = () => {
185     this.setState({ isEditing: false, editPost: null });
186 };
187
188 finishEditHandler = postData => {
189     this.setState({
190         editLoading: true
191     });
192     const formData = new FormData();
193     formData.append('title', postData.title);
194     formData.append('content', postData.content);
195     formData.append('image', postData.image);
196     let url = 'http://localhost:8080/feed/post';
197     let method = 'POST';
198     if (this.state.editPost) {
199         url = 'http://localhost:8080/feed/post/' + this.state.editPost._id;
200         method = 'PUT';
201     }
202
203     fetch(url, {
204         method: method,
205         body: formData,
206         headers: {
207             Authorization: 'Bearer ' + this.props.token
208         }
209     })
210     .then(res => {
211         if (res.status !== 200 && res.status !== 201) {
212             throw new Error('Creating or editing a post failed!');
213         }
214         return res.json();
215     })
216     .then(resData => {
217         console.log(resData);
218         const post = {
219             _id: resData.post._id,
220             title: resData.post.title,
221             content: resData.post.content,
222             creator: resData.post.creator,
223             createdAt: resData.post.createdAt
224         };
225         this.setState(prevState => {
226             /**i now no longer updated here
227              * but i admit the event to all connect clients
228              * including to clients who did well sent that update
229              * and therefore i can also set up my listener and new state
230              * with this new update post function 'updatePost'
231              */
232             return {
233                 isEditing: false,
234                 editPost: null,

```

```

235         editLoading: false
236     };
237 });
238 })
239 .catch(err => {
240     console.log(err);
241     this.setState({
242         isEditing: false,
243         editPost: null,
244         editLoading: false,
245         error: err
246     });
247 });
248 };
249
250 statusInputChangeHandler = (input, value) => {
251     this.setState({ status: value });
252 };
253
254 deletePostHandler = postId => {
255     this.setState({ postsLoading: true });
256     fetch('http://localhost:8080/feed/post/' + postId, {
257         method: 'DELETE',
258         headers: {
259             Authorization: 'Bearer ' + this.props.token
260         }
261     })
262     .then(res => {
263         if (res.status !== 200 && res.status !== 201) {
264             throw new Error('Deleting a post failed!');
265         }
266         return res.json();
267     })
268     .then(resData => {
269         console.log(resData);
270         this.setState(prevState => {
271             const updatedPosts = prevState.posts.filter(p => p._id !== postId);
272             return { posts: updatedPosts, postsLoading: false };
273         });
274     })
275     .catch(err => {
276         console.log(err);
277         this.setState({ postsLoading: false });
278     });
279 };
280
281 errorHandler = () => {
282     this.setState({ error: null });
283 };
284
285 catchError = error => {
286     this.setState({ error: error });
287 };
288
289 render() {
290     return (

```



```

291 <Fragment>
292   <ErrorHandler error={this.state.error} onHandle={this.errorHandler} />
293   <FeedEdit
294     editing={this.state.isEditing}
295     selectedPost={this.state.editPost}
296     loading={this.state.editLoading}
297     onCancelEdit={this.cancelEditHandler}
298     onFinishEdit={this.finishEditHandler}
299   />
300   <section className="feed__status">
301     <form onSubmit={this.statusUpdateHandler}>
302       <Input
303         type="text"
304         placeholder="Your status"
305         control="input"
306         onChange={this.statusInputChangeHandler}
307         value={this.state.status}
308       />
309       <Button mode="flat" type="submit">
310         Update
311       </Button>
312     </form>
313   </section>
314   <section className="feed__control">
315     <Button mode="raised" design="accent" onClick={this.newPostHandler}>
316       New Post
317     </Button>
318   </section>
319   <section className="feed">
320     {this.state.postsLoading && (
321       <div style={{ textAlign: 'center', marginTop: '2rem' }}>
322         <Loader />
323       </div>
324     )}
325     {this.state.posts.length <= 0 && !this.state.postsLoading ? (
326       <p style={{ textAlign: 'center' }}>No posts found.</p>
327     ) : null}
328     {!this.state.postsLoading && (
329       <Paginator
330         onPrevious={this.loadPosts.bind(this, 'previous')}
331         onNext={this.loadPosts.bind(this, 'next')}
332         lastPage={Math.ceil(this.state.totalPosts / 2)}
333         currentPage={this.state.postPage}
334       >
335         {this.state.posts.map(post => (
336           <Post
337             key={post._id}
338             id={post._id}
339             author={post.creator.name}
340             date={new Date(post.createdAt).toLocaleDateString('en-US')}
341             title={post.title}
342             image={post.imageUrl}
343             content={post.content}
344             onStartEdit={this.startEditPostHandler.bind(this, post._id)}
345             onDelete={this.deletePostHandler.bind(this, post._id)}
346           />

```

```

347     })}
348     </Paginator>
349   })}
350 </section>
351 </Fragment>
352 );
353 }
354 }
355
356 export default Feed;

```

## \* Chapter 411: Sorting Correctly

1. update

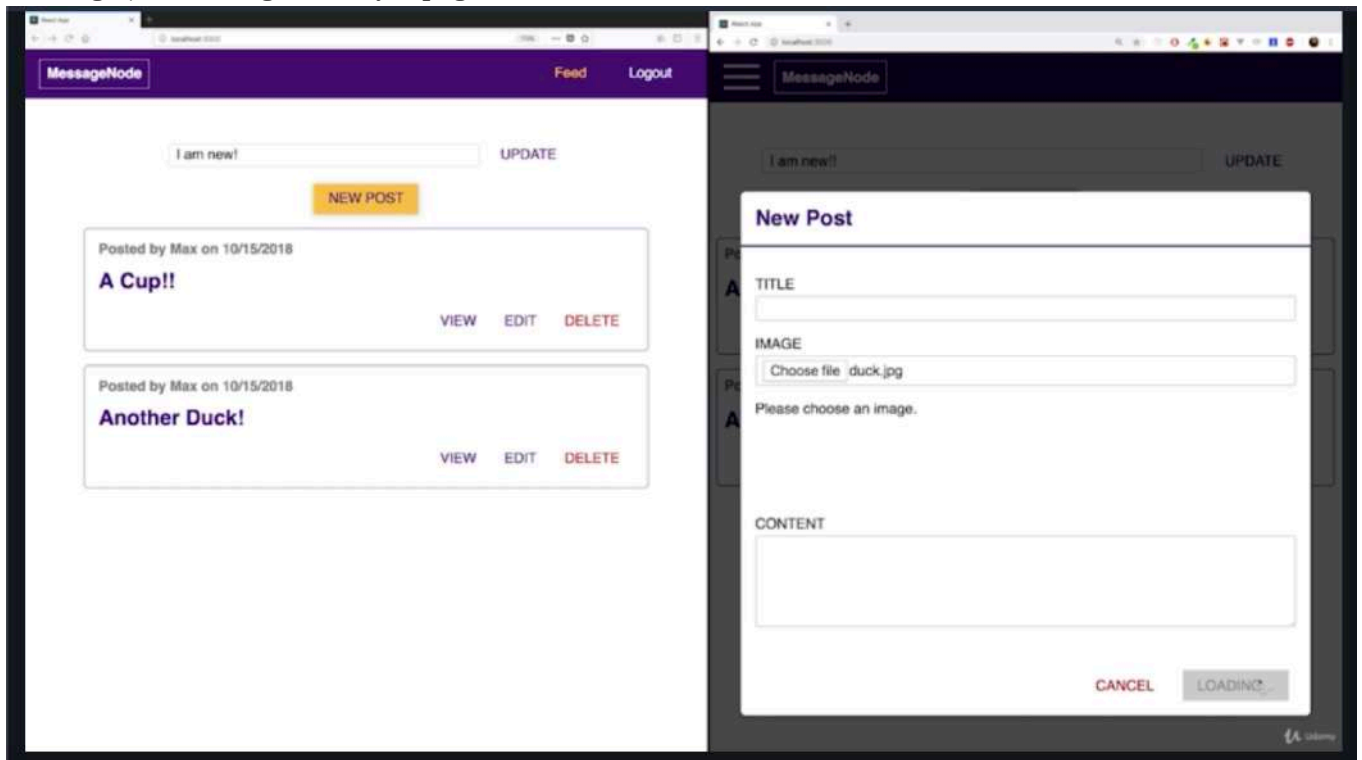
- ./controllers/feed.js(b)

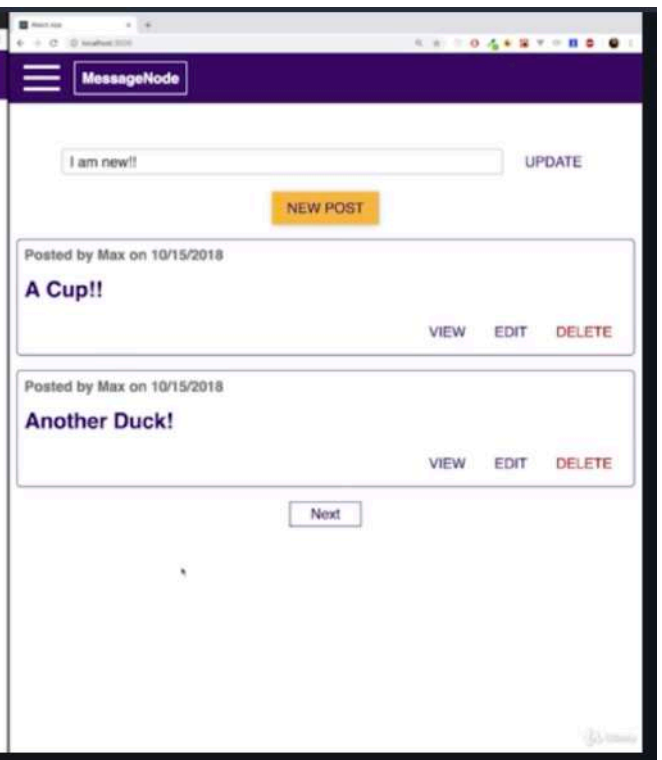
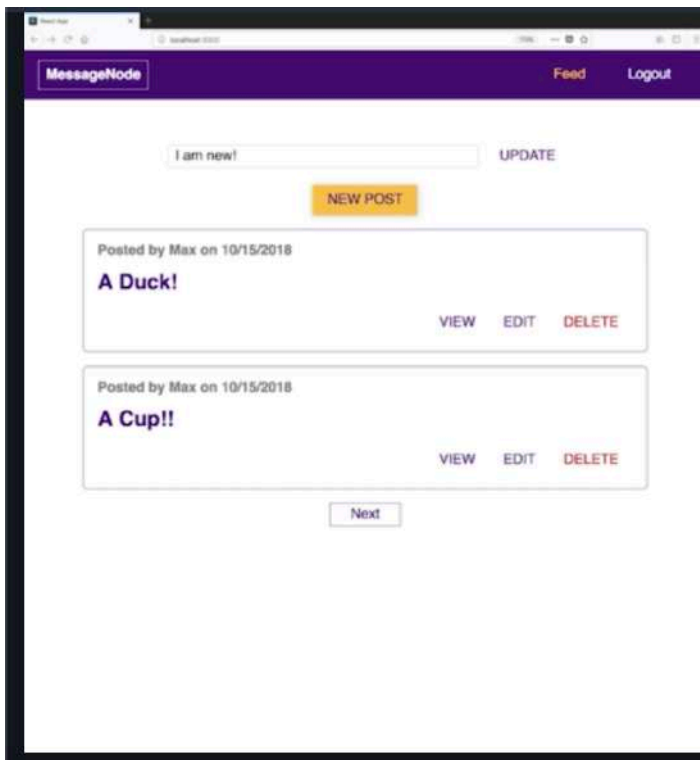
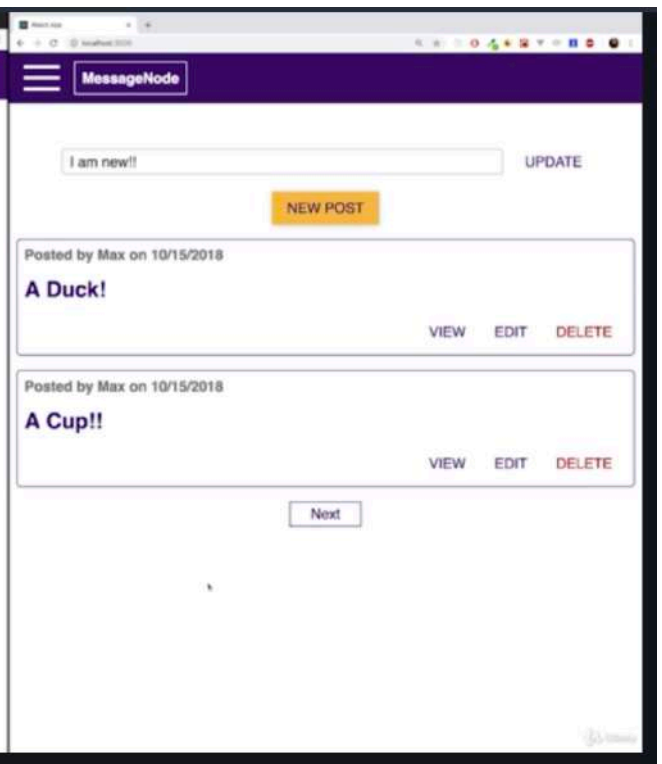
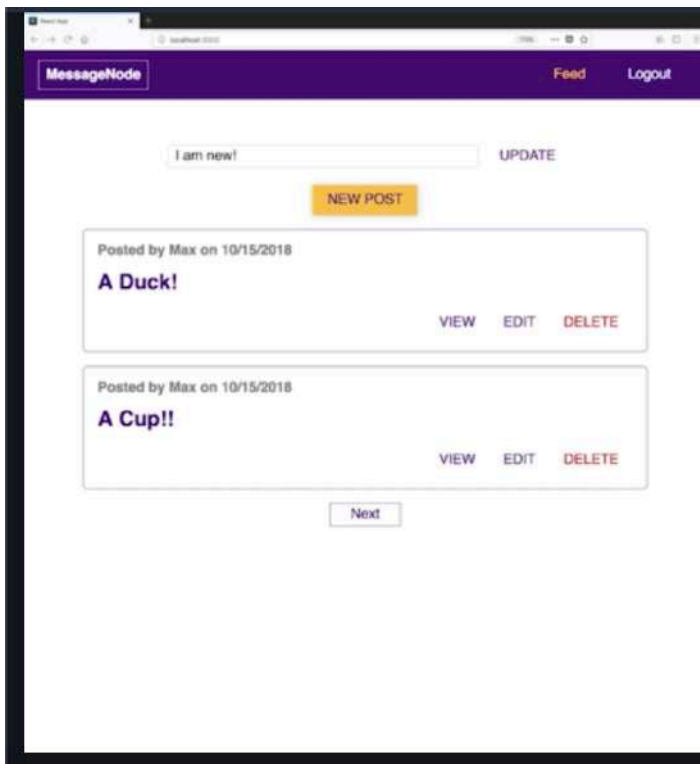


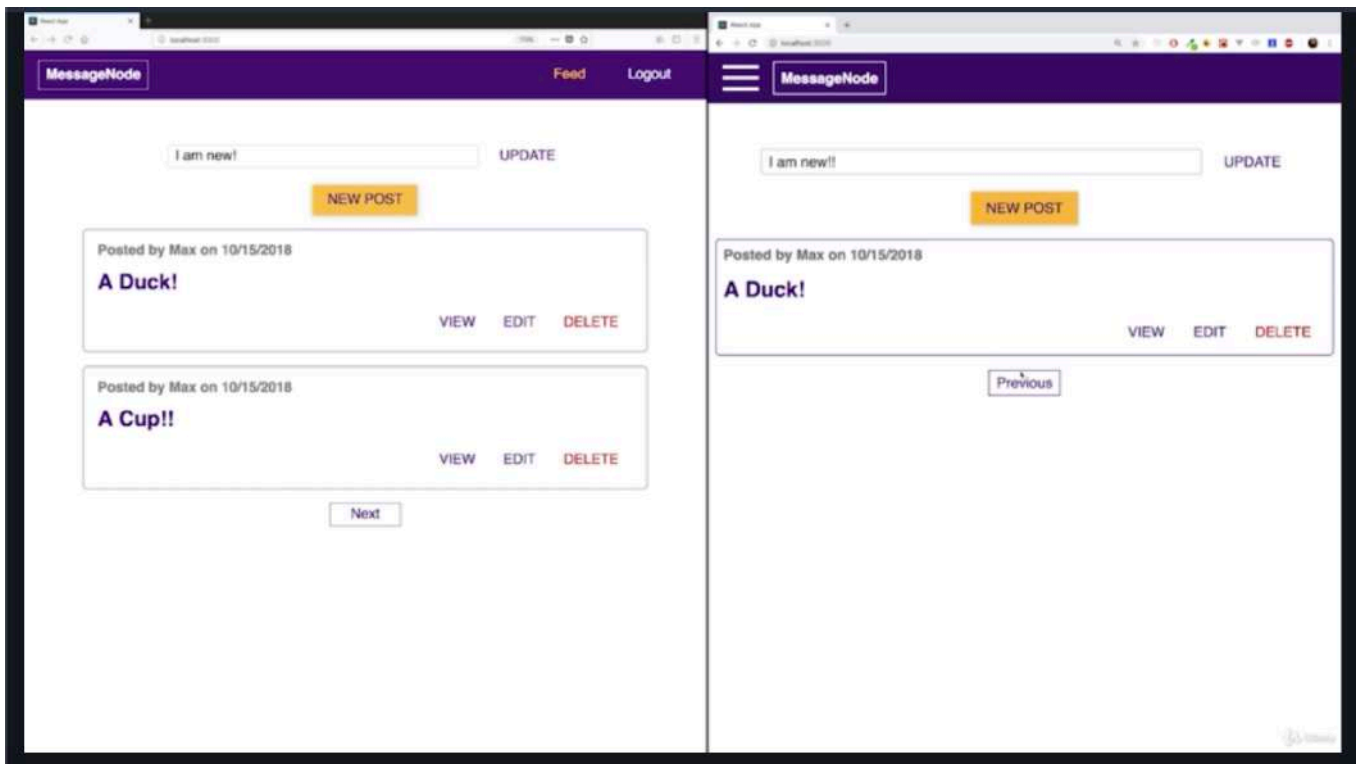








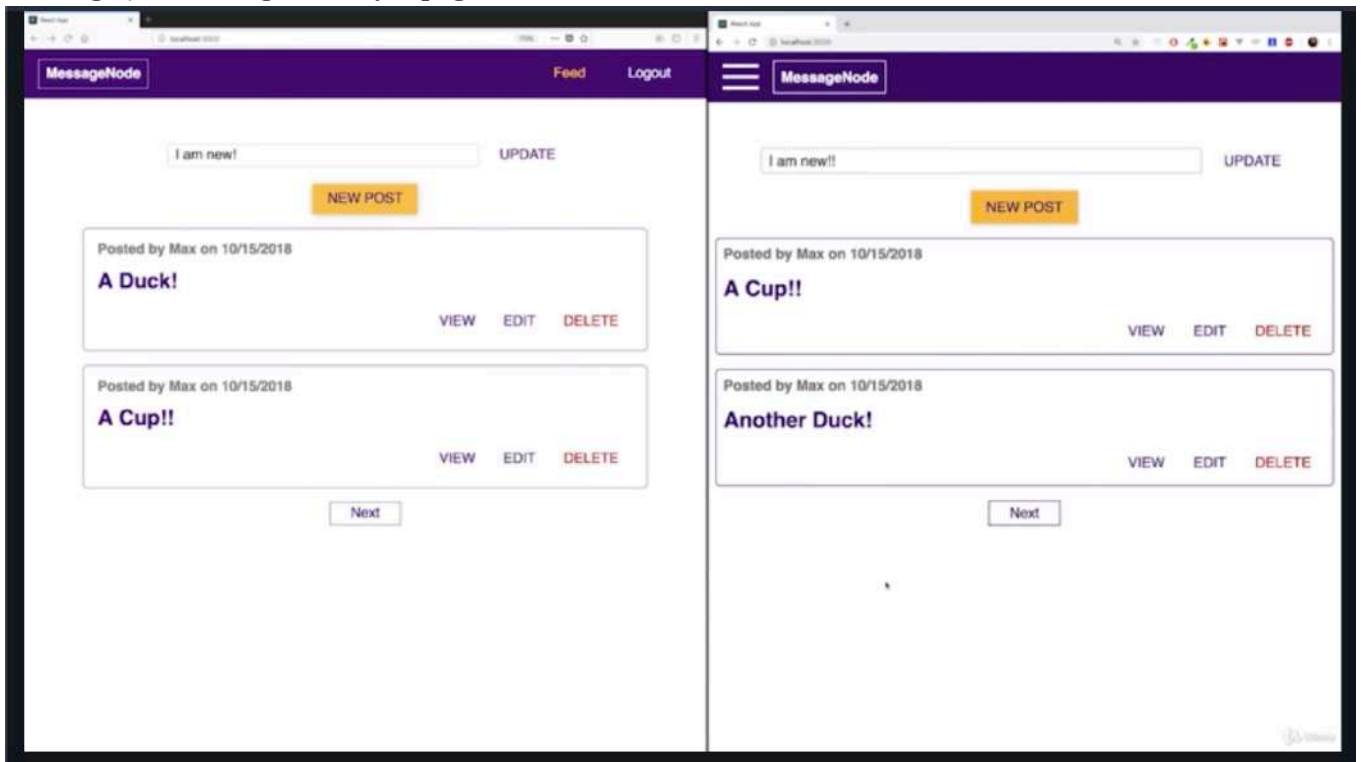


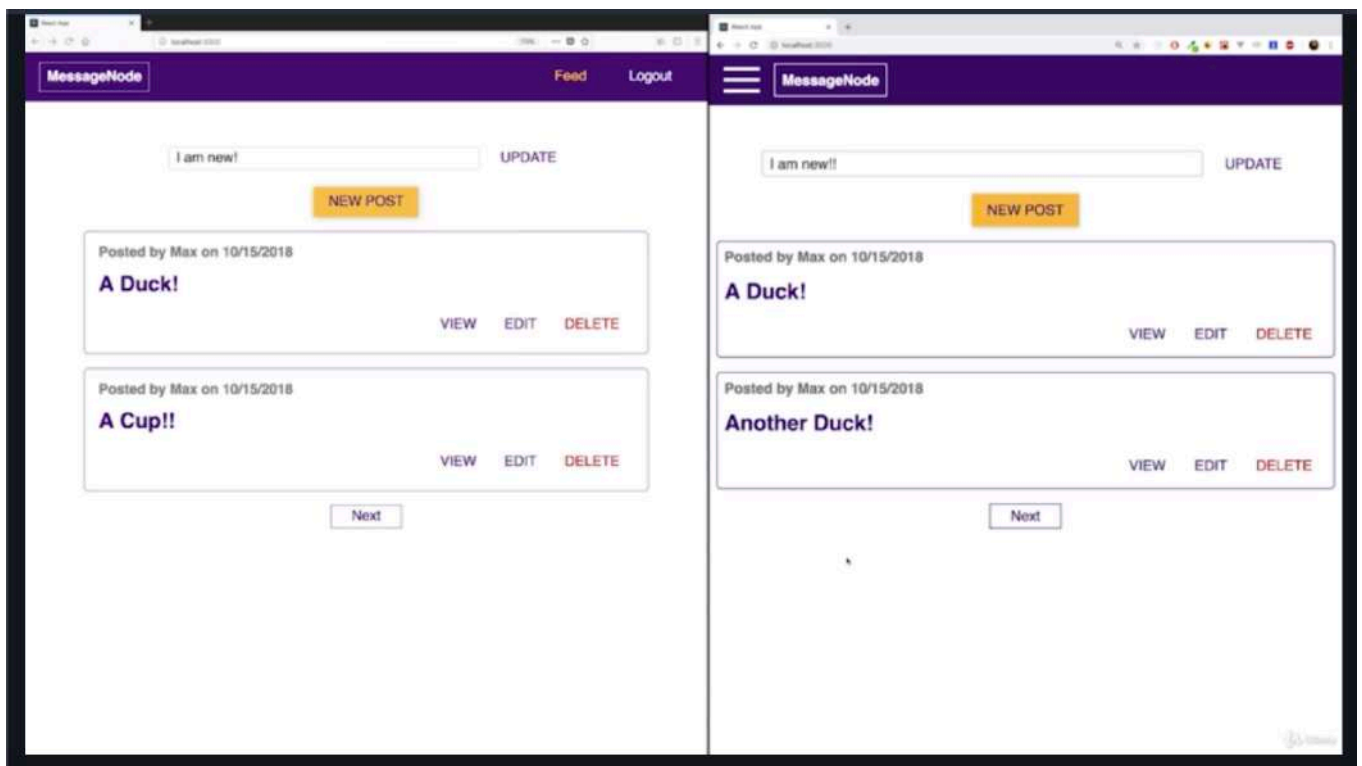


- if we create 'A Duck!' and reload then go to end. so we are gonna fix this with 'sort()'









- now we fix that. latest post is the first.

```

1  ../controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const io = require('../socket');
9  const Post = require('../models/post');
10 const User = require('../models/user');
11
12 exports.getPosts = async (req, res, next) => {
13   const currentPage = req.query.page || 1;
14   const perPage = 2;
15   try {
16     const totalItems = await Post.find().countDocuments();
17     const posts = await Post.find()
18       .populate('creator')
19       .sort({ createdAt: -1 })
20       .skip((currentPage - 1) * perPage)
21       .limit(perPage);
22
23     res.status(200).json({
24       message: 'Fetched posts successfully.',
25       posts: posts,
26       totalItems: totalItems
27     });
28   } catch (err) {
29     if (!err.statusCode) {
30       err.statusCode = 500;
31     }
32     next(err);
33   }

```

```

34 };
35
36 exports.createPost = async (req, res, next) => {
37   const errors = validationResult(req);
38   if (!errors.isEmpty()) {
39     const error = new Error('Validation failed, entered data is incorrect.');
```

error.statusCode = 422;

```

41     throw error;
42   }
43   if (!req.file) {
44     const error = new Error('No image provided.');
```

error.statusCode = 422;

```

46     throw error;
47   }
48   const imageUrl = req.file.path;
49   const title = req.body.title;
50   const content = req.body.content;
51   const post = new Post({
52     title: title,
53     content: content,
54     imageUrl: imageUrl,
55     creator: req.userId
56   });
57   try {
58     await post.save();
59     const user = await User.findById(req.userId);
60     user.posts.push(post);
61     await user.save();
62     io.getIO().emit('posts', {
63       action: 'create',
64       post: {
65         ...post._doc,
66         creator: { _id: req.userId, name: user.name } }
67   });
68     res.status(201).json({
69       message: 'Post created successfully!',
70       post: post,
71       creator: { _id: user._id, name: user.name }
72   });
73   } catch (err) {
74     if (!err.statusCode) {
75       err.statusCode = 500;
76     }
77     next(err);
78   }
79 };
80
81 exports.getPost = async (req, res, next) => {
82   const postId = req.params.postId;
83   const post = await Post.findById(postId);
84   try {
85     if (!post) {
86       const error = new Error('Could not find post.');
```

error.statusCode = 404;

```

88       throw error;
89     }

```

```

90     res.status(200).json({ message: 'Post fetched.', post: post });
91 } catch (err) {
92     if (!err.statusCode) {
93         err.statusCode = 500;
94     }
95     next(err);
96 }
97 };
98
99 exports.updatePost = async (req, res, next) => {
100     const postId = req.params.postId;
101     const errors = validationResult(req);
102     if (!errors.isEmpty()) {
103         const error = new Error('Validation failed, entered data is incorrect.');
```

104 error.statusCode = 422;

105 throw error;

106 }

107 const title = req.body.title;

108 const content = req.body.content;

109 let imageUrl = req.body.image;

110 if (req.file) {

111 imageUrl = req.file.path;

112 }

113 if (!imageUrl) {

114 const error = new Error('No file picked.');

115 error.statusCode = 422;

116 throw error;

117 }

118 try {

119 const post = await Post.findById(postId).populate('creator');

120 if (!post) {

121 const error = new Error('Could not find post.');

122 error.statusCode = 404;

123 throw error;

124 }

125 if (post.creator.\_id.toString() !== req.userId) {

126 const error = new Error('Not authorized!');

127 error.statusCode = 403;

128 throw error;

129 }

130 if (imageUrl !== post.imageUrl) {

131 clearImage(post.imageUrl);

132 }

133 post.title = title;

134 post.imageUrl = imageUrl;

135 post.content = content;

136 const result = await post.save();

137 io.getIO().emit('posts', { action: 'update', post: result });

138 res.status(200).json({ message: 'Post updated!', post: result });

139 } catch (err) {

140 if (!err.statusCode) {

141 err.statusCode = 500;

142 }

143 next(err);

144 }

145 };

```

146
147 exports.deletePost = async (req, res, next) => {
148   const postId = req.params.postId;
149   try {
150     const post = await Post.findById(postId);
151
152     if (!post) {
153       const error = new Error('Could not find post.');
```

```

154       error.statusCode = 404;
155       throw error;
156     }
157     if (post.creator.toString() !== req.userId) {
158       const error = new Error('Not authorized!');
```

```

159       error.statusCode = 403;
160       throw error;
161     }
162     // Check logged in user
163     clearImage(post.imageUrl);
164     await Post.findByIdAndRemove(postId);
165
166     const user = await User.findById(req.userId);
167     user.posts.pull(postId);
168     await user.save();
169
170     res.status(200).json({ message: 'Deleted post.' });
171   } catch (err) {
172     if (!err.statusCode) {
173       err.statusCode = 500;
174     }
175     next(err);
176   }
177 };
178
179 const clearImage = filePath => {
180   filePath = path.join(__dirname, '..', filePath);
181   fs.unlink(filePath, err => console.log(err));
182 };
183

```

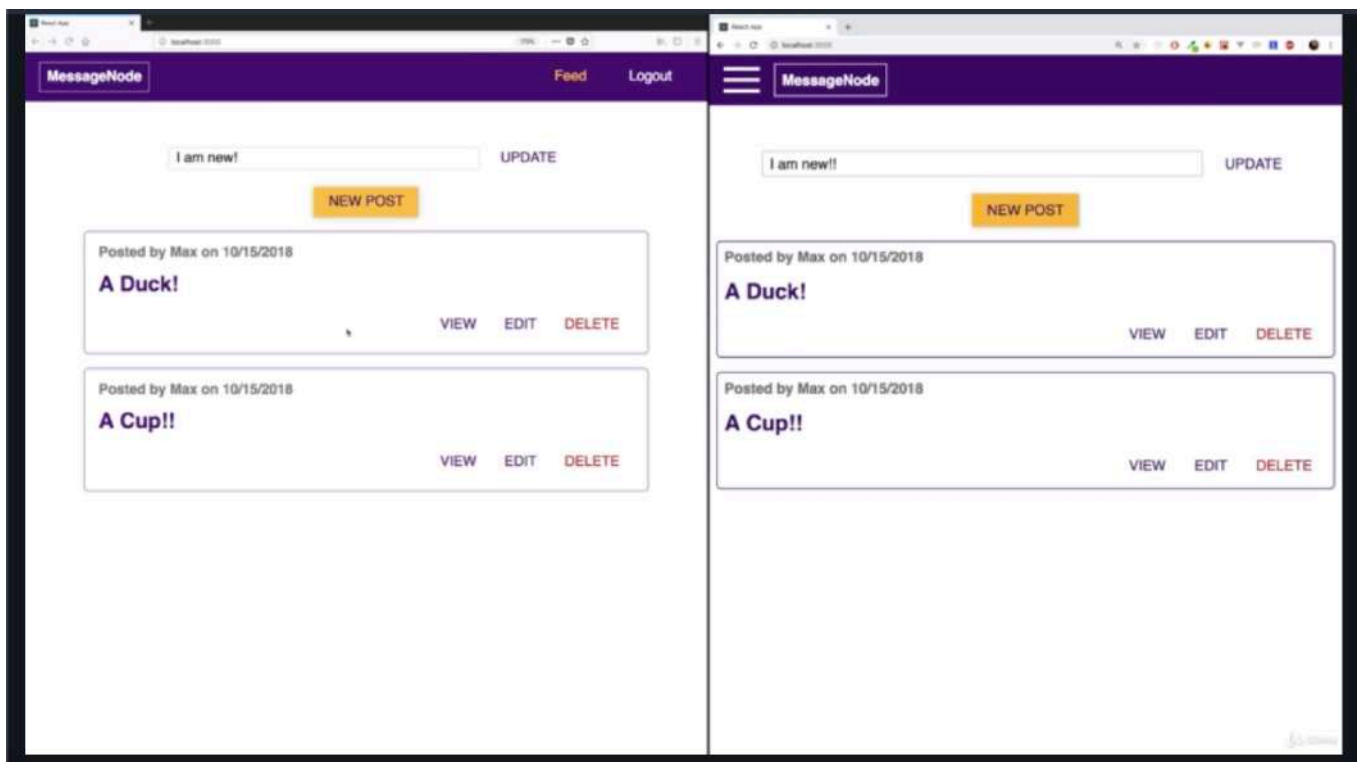
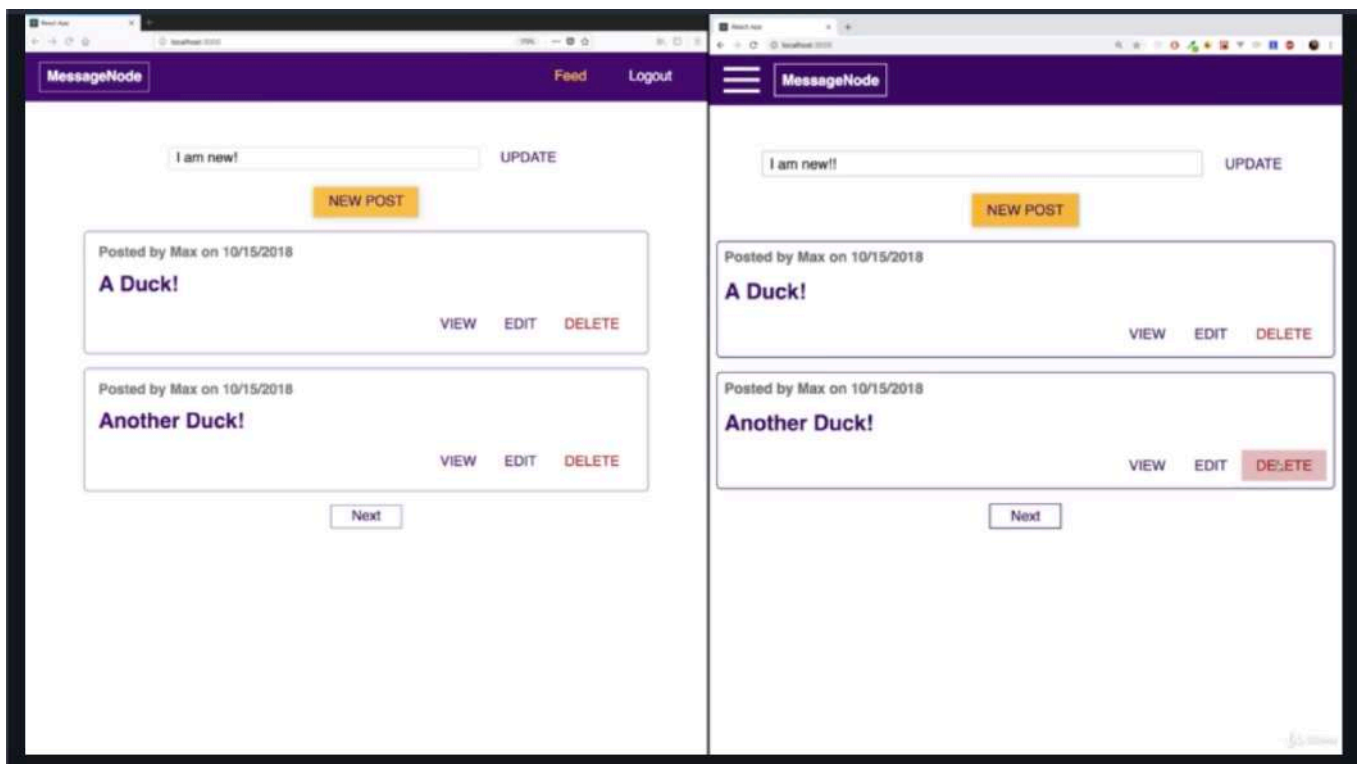
## \* Chapter 412: Deleting Posts Across Clients

1. update
  - ./controllers/feed.js(b)
  - ./src/pages/Feed/Feed.js(f)









- if we click DELETE, it reloads and is removed

```

1  ../controllers/feed.js(b)
2
3  const fs = require('fs');
4  const path = require('path');
5
6  const { validationResult } = require('express-validator/check');
7
8  const io = require('../socket');
9  const Post = require('../models/post');
10 const User = require('../models/user');
11
12 exports.getPosts = async (req, res, next) => {

```

```

13 const currentPage = req.query.page || 1;
14 const perPage = 2;
15 try {
16     const totalItems = await Post.find().countDocuments();
17     const posts = await Post.find()
18         .populate('creator')
19         .sort({ createdAt: -1 })
20         .skip((currentPage - 1) * perPage)
21         .limit(perPage);
22
23     res.status(200).json({
24         message: 'Fetched posts successfully.',
25         posts: posts,
26         totalItems: totalItems
27     });
28 } catch (err) {
29     if (!err.statusCode) {
30         err.statusCode = 500;
31     }
32     next(err);
33 }
34 };
35
36 exports.createPost = async (req, res, next) => {
37     const errors = validationResult(req);
38     if (!errors.isEmpty()) {
39         const error = new Error('Validation failed, entered data is incorrect.');
```

error.statusCode = 422;

throw error;

}

if (!req.file) {

const error = new Error('No image provided.');

error.statusCode = 422;

throw error;

}

const imageUrl = req.file.path;

const title = req.body.title;

const content = req.body.content;

const post = new Post({

title: title,

content: content,

imageUrl: imageUrl,

creator: req.userId

});

try {

await post.save();

const user = await User.findById(req.userId);

user.posts.push(post);

await user.save();

io.getIO().emit('posts', {

action: 'create',

post: {

...post.\_doc,

creator: { \_id: req.userId, name: user.name } }

});

res.status(201).json({

```

69     message: 'Post created successfully!',
70     post: post,
71     creator: { _id: user._id, name: user.name }
72   });
73 } catch (err) {
74   if (!err.statusCode) {
75     err.statusCode = 500;
76   }
77   next(err);
78 }
79 };
80
81 exports.getPost = async (req, res, next) => {
82   const postId = req.params.postId;
83   const post = await Post.findById(postId);
84   try {
85     if (!post) {
86       const error = new Error('Could not find post.');
```

```

87       error.statusCode = 404;
88       throw error;
89     }
90     res.status(200).json({ message: 'Post fetched.', post: post });
91   } catch (err) {
92     if (!err.statusCode) {
93       err.statusCode = 500;
94     }
95     next(err);
96   }
97 };
98
99 exports.updatePost = async (req, res, next) => {
100   const postId = req.params.postId;
101   const errors = validationResult(req);
102   if (!errors.isEmpty()) {
103     const error = new Error('Validation failed, entered data is incorrect.');
```

```

104     error.statusCode = 422;
105     throw error;
106   }
107   const title = req.body.title;
108   const content = req.body.content;
109   let imageUrl = req.body.image;
110   if (req.file) {
111     imageUrl = req.file.path;
112   }
113   if (!imageUrl) {
114     const error = new Error('No file picked.');
```

```

115     error.statusCode = 422;
116     throw error;
117   }
118   try {
119     const post = await Post.findById(postId).populate('creator');
```

```

120     if (!post) {
121       const error = new Error('Could not find post.');
```

```

122       error.statusCode = 404;
123       throw error;
124     }

```

```

125     if (post.creator._id.toString() !== req.userId) {
126         const error = new Error('Not authorized!');
127         error.statusCode = 403;
128         throw error;
129     }
130     if (imageUrl !== post.imageUrl) {
131         clearImage(post.imageUrl);
132     }
133     post.title = title;
134     post.imageUrl = imageUrl;
135     post.content = content;
136     const result = await post.save();
137     io.getIO().emit('posts', { action: 'update', post: result });
138     res.status(200).json({ message: 'Post updated!', post: result });
139 } catch (err) {
140     if (!err.statusCode) {
141         err.statusCode = 500;
142     }
143     next(err);
144 }
145 };
146
147 exports.deletePost = async (req, res, next) => {
148     const postId = req.params.postId;
149     try {
150         const post = await Post.findById(postId);
151
152         if (!post) {
153             const error = new Error('Could not find post. ');
154             error.statusCode = 404;
155             throw error;
156         }
157         if (post.creator.toString() !== req.userId) {
158             const error = new Error('Not authorized!');
159             error.statusCode = 403;
160             throw error;
161         }
162         // Check logged in user
163         clearImage(post.imageUrl);
164         await Post.findByIdAndRemove(postId);
165
166         const user = await User.findById(req.userId);
167         user.posts.pull(postId);
168         await user.save();
169         io.getIO().emit('posts', { action: 'delete', post: postId });
170         res.status(200).json({ message: 'Deleted post.' });
171     } catch (err) {
172         if (!err.statusCode) {
173             err.statusCode = 500;
174         }
175         next(err);
176     }
177 };
178
179 const clearImage = filePath => {
180     filePath = path.join(__dirname, '..', filePath);

```

```
181 fs.unlink(filePath, err => console.log(err));
182 };
183
```

```
1  //./src/pages/Feed/Feed.js(f)
2
3  import React, { Component, Fragment } from 'react';
4  import openSocket from 'socket.io-client';
5
6  import Post from '../components/Feed/Post/Post';
7  import Button from '../components/Button/Button';
8  import FeedEdit from '../components/Feed/FeedEdit/FeedEdit';
9  import Input from '../components/Form/Input/Input';
10 import Paginator from '../components/Paginator/Paginator';
11 import Loader from '../components/Loader/Loader';
12 import ErrorHandler from '../components/ErrorHandler/ErrorHandler';
13 import './Feed.css';
14
15 class Feed extends Component {
16   state = {
17     isEditing: false,
18     posts: [],
19     totalPosts: 0,
20     editPost: null,
21     status: '',
22     postPage: 1,
23     postsLoading: true,
24     editLoading: false
25   };
26
27   componentDidMount() {
28     fetch('http://localhost:8080/auth/status', {
29       headers: {
30         Authorization: 'Bearer ' + this.props.token
31       }
32     })
33       .then(res => {
34         if (res.status !== 200) {
35           throw new Error('Failed to fetch user status.');
```

```

54     });
55 }
56
57 addPost = post => {
58     this.setState(prevState => {
59         const updatedPosts = [...prevState.posts];
60         if (prevState.postPage === 1) {
61             if (prevState.posts.length >= 2) {
62                 updatedPosts.pop();
63             }
64             updatedPosts.unshift(post);
65         }
66         return {
67             posts: updatedPosts,
68             totalPosts: prevState.totalPosts + 1
69         };
70     });
71 };
72
73 updatePost = post => {
74     this.setState(prevState => {
75         const updatedPosts = [...prevState.posts];
76         const updatedPostIndex = updatedPosts.findIndex(p => p._id === post._id);
77         if (updatedPostIndex > -1) {
78             updatedPosts[updatedPostIndex] = post;
79         }
80         return {
81             posts: updatedPosts
82         };
83     });
84 }
85
86 loadPosts = direction => {
87     if (direction) {
88         this.setState({ postsLoading: true, posts: [] });
89     }
90     let page = this.state.postPage;
91     if (direction === 'next') {
92         page++;
93         this.setState({ postPage: page });
94     }
95     if (direction === 'previous') {
96         page--;
97         this.setState({ postPage: page });
98     }
99     fetch('http://localhost:8080/feed/posts?page=' + page, {
100         headers: {
101             Authorization: 'Bearer ' + this.props.token
102         }
103     })
104     .then(res => {
105         if (res.status !== 200) {
106             throw new Error('Failed to fetch posts.');
```

```

110 .then(resData => {
111   this.setState({
112     posts: resData.posts.map(post => {
113       return {
114         ...post,
115         imagePath: post.imageUrl
116       };
117     }),
118     totalPosts: resData.totalItems,
119     postsLoading: false
120   });
121 })
122 .catch(this.catchError);
123 };
124
125 statusUpdateHandler = event => {
126   event.preventDefault();
127   fetch('http://localhost:8080/auth/status', {
128     method: 'PATCH',
129     headers: {
130       Authorization: 'Bearer ' + this.props.token,
131       'Content-Type': 'application/json'
132     },
133     body: JSON.stringify({
134       status: this.state.status
135     })
136   })
137   .then(res => {
138     if (res.status !== 200 && res.status !== 201) {
139       throw new Error("Can't update status!");
140     }
141     return res.json();
142   })
143   .then(resData => {
144     console.log(resData);
145   })
146   .catch(this.catchError);
147 };
148
149 newPostHandler = () => {
150   this.setState({ isEditing: true });
151 };
152
153 startEditPostHandler = postId => {
154   this.setState(prevState => {
155     const loadedPost = { ...prevState.posts.find(p => p._id === postId) };
156
157     return {
158       isEditing: true,
159       editPost: loadedPost
160     };
161   });
162 };
163
164 cancelEditHandler = () => {
165   this.setState({ isEditing: false, editPost: null });

```

```

166 };
167
168 finishEditHandler = postData => {
169     this.setState({
170         editLoading: true
171     });
172     const formData = new FormData();
173     formData.append('title', postData.title);
174     formData.append('content', postData.content);
175     formData.append('image', postData.image);
176     let url = 'http://localhost:8080/feed/post';
177     let method = 'POST';
178     if (this.state.editPost) {
179         url = 'http://localhost:8080/feed/post/' + this.state.editPost._id;
180         method = 'PUT';
181     }
182
183     fetch(url, {
184         method: method,
185         body: formData,
186         headers: {
187             Authorization: 'Bearer ' + this.props.token
188         }
189     })
190     .then(res => {
191         if (res.status !== 200 && res.status !== 201) {
192             throw new Error('Creating or editing a post failed!');
193         }
194         return res.json();
195     })
196     .then(resData => {
197         console.log(resData);
198         const post = {
199             _id: resData.post._id,
200             title: resData.post.title,
201             content: resData.post.content,
202             creator: resData.post.creator,
203             createdAt: resData.post.createdAt
204         };
205         this.setState(prevState => {
206             return {
207                 isEditing: false,
208                 editPost: null,
209                 editLoading: false
210             };
211         });
212     })
213     .catch(err => {
214         console.log(err);
215         this.setState({
216             isEditing: false,
217             editPost: null,
218             editLoading: false,
219             error: err
220         });
221     });

```



```

222 };
223
224 statusInputChangeHandler = (input, value) => {
225   this.setState({ status: value });
226 };
227
228 deletePostHandler = postId => {
229   this.setState({ postsLoading: true });
230   fetch('http://localhost:8080/feed/post/' + postId, {
231     method: 'DELETE',
232     headers: {
233       Authorization: 'Bearer ' + this.props.token
234     }
235   })
236     .then(res => {
237       if (res.status !== 200 && res.status !== 201) {
238         throw new Error('Deleting a post failed!');
239       }
240       return res.json();
241     })
242     .then(resData => {
243       console.log(resData);
244       this.loadPosts();
245       /*
246       this.setState(prevState => {
247         const updatedPosts = prevState.posts.filter(p => p._id !== postId);
248         return { posts: updatedPosts, postsLoading: false };
249       });
250       */
251     })
252     .catch(err => {
253       console.log(err);
254       this.setState({ postsLoading: false });
255     });
256 };
257
258 errorHandler = () => {
259   this.setState({ error: null });
260 };
261
262 catchError = error => {
263   this.setState({ error: error });
264 };
265
266 render() {
267   return (
268     <Fragment>
269       <ErrorHandler error={this.state.error} onHandle={this.errorHandler} />
270       <FeedEdit
271         editing={this.state.isEditing}
272         selectedPost={this.state.editPost}
273         loading={this.state.editLoading}
274         onCancelEdit={this.cancelEditHandler}
275         onFinishEdit={this.finishEditHandler}
276       />
277       <section className="feed__status">

```

```

278 <form onSubmit={this.statusUpdateHandler}>
279   <Input
280     type="text"
281     placeholder="Your status"
282     control="input"
283     onChange={this.statusInputChangeHandler}
284     value={this.state.status}
285   />
286   <Button mode="flat" type="submit">
287     Update
288   </Button>
289 </form>
290 </section>
291 <section className="feed__control">
292   <Button mode="raised" design="accent" onClick={this.newPostHandler}>
293     New Post
294   </Button>
295 </section>
296 <section className="feed">
297   {this.state.postsLoading && (
298     <div style={{ textAlign: 'center', marginTop: '2rem' }}>
299       <Loader />
300     </div>
301   )}
302   {this.state.posts.length <= 0 && !this.state.postsLoading ? (
303     <p style={{ textAlign: 'center' }}>No posts found.</p>
304   ) : null}
305   {!this.state.postsLoading && (
306     <Paginator
307       onPrevious={this.loadPosts.bind(this, 'previous')}
308       onNext={this.loadPosts.bind(this, 'next')}
309       lastPage={Math.ceil(this.state.totalPosts / 2)}
310       currentPage={this.state.postPage}
311     >
312       {this.state.posts.map(post => (
313         <Post
314           key={post._id}
315           id={post._id}
316           author={post.creator.name}
317           date={new Date(post.createdAt).toLocaleDateString('en-US')}
318           title={post.title}
319           image={post.imageUrl}
320           content={post.content}
321           onStartEdit={this.startEditPostHandler.bind(this, post._id)}
322           onDelete={this.deletePostHandler.bind(this, post._id)}
323         />
324       ))}
325     </Paginator>
326   )}
327 </section>
328 </Fragment>
329 );
330 }
331 }
332
333 export default Feed;

```

