

**Lesson 5-Building RESTful APIs with MERN Stack and Redux Toolkit**

**Activity 8 –PostIT App – Like and Dislike**

The objectives of this activity are to:

1. Implement the like and dislike feature.
2. Use React Icons.

**IMPLEMENTING THE LIKE AND DISLIKE**

**SERVER-SIDE**

1. Create a PUT API, we are PUT method because every time the user clicks the like, we are going to update the count value of the likes. In order to update, we need to have the postId. Thus, it should be passed as a parameter in the URL. Extract the ID of the post from the URL, which is used to identify the specific post.

app.put("/likePost/:postId/", async (req, res) => {

    const postId = req.params.postId; //Extract the ID of the post from the URL

    const userId = req.body.userId;

    try {

    } catch (err) {

      console.error(err);

      res.status(500).json({ error: "An error occurred" });

    }

  });

1. Use the **findOneAndUpdate** method of Mongoose to find the post and update accordingly.

//PUT API - likePost

app.put("/likePost/:postId/", async (req, res) => {

    const postId = req.params.postId;

    const userId = req.body.userId;

      try {

      //search the postId if it exists

      const postToUpdate = await PostModel.findOne({ \_id: postId });

      if (!postToUpdate) {

        return res.status(404).json({ msg: "Post not found." });

      }

      //Search the user Id from the array of users who liked the post.

      const userIndex = postToUpdate.likes.users.indexOf(userId);

      //indexOf method returns the index of the first occurrence of a specified value in an array.

      //If the value is not found, it returns -1.

      //This code will toogle from like to unlike

      if (userIndex !== -1) {

        // User has already liked the post, so unlike it

        const udpatedPost = await PostModel.findOneAndUpdate(

          { \_id: postId },

          {

            $inc: { "likes.count": -1 }, // Decrement the like count $inc and $pull are update operators

            $pull: { "likes.users": userId }, // Remove userId from the users array

          },

          { new: true } // Return the modified document

        );

        res.json({ post: udpatedPost, msg: "Post unliked." });

      } else {

        // User hasn't liked the post, so like it

        const updatedPost = await PostModel.findOneAndUpdate(

          { \_id: postId },

          {

            $inc: { "likes.count": 1 }, // Increment the like count

            $addToSet: { "likes.users": userId }, // Add userId to the users array if not already present

          },

          { new: true } // Return the modified document

        );

        res.json({ post: updatedPost, msg: "Post liked." });

      }

    } catch (err) {

      console.error(err);

      res.status(500).json({ error: "An error occurred" });

    }

  });

**$inc, $pull** and **$addToSet** are update operators used in MongoDB's update operations to modify documents in a collection. They are part of the update document that specifies how the document should be updated.

**CLIENT SIDE**

1. Create a new thunk in the Post slice in src/Features/PostsSlice.js.

export const likePost = createAsyncThunk("posts/likePost", async (postData) => {

    try {

      //Pass along the URL the postId

      const response = await axios.put(

        `http://localhost:3001/likePost/${postData.postId}`,

        {

          userId: postData.userId,

        }

      );

      const post = response.data.post;

      return post;

    } catch (error) {

      console.log(error);

    }

  });

1. Create the extrareducer for the likePost thunk function.

extraReducers: (builder) => {

    builder

      …..

      .addCase(likePost.pending, (state) => {

        state.status = "loading";

      })

      .addCase(likePost.fulfilled, (state, action) => {

        state.status = "succeeded";

        //Search the post id from the posts state

        const updatedPostIndex = state.posts.findIndex(

          (post) => post.\_id === action.payload.\_id

        );

//If found, update the likes property of the found post to the current value of the likes

        if (updatedPostIndex !== -1) {

          state.posts[updatedPostIndex].likes = action.payload.likes;

        }

      })

      .addCase(likePost.rejected, (state, action) => {

        state.status = "failed";

        state.error = action.error.message;

      });

  },

1. Import the likePost from the PostSlice.

import { likePost } from "../Features/PostSlice";

1. Dispatch the **likePost** action in the Posts in the component **src/Components/Posts.js**. Make a function **handleLikePost()** this will be invoked when the user clicks the like hyperlink.

const Posts = () => {

    …

      const handleLikePost = (postId) => {

        const postData = {

          postId: postId,

          userId: userId,

        };

        dispatch(likePost(postData));

        navigate("/home");

      };

1. Using the React Icons library:
   1. Install the library in your client folder: npm install react-icon
   2. After installation, you can import the icons you need in your components. For example:

import { FaThumbsUp } from "react-icons/fa6";

* 1. Browse the website of React Icons documentation.  
     <https://react-icons.github.io/react-icons/>

1. In the table displaying the Posts, all the **handleLikePost** function when the user clicks the thumbs up icon and pass the postId as argument.

<tbody>

{posts.map((post) => (

  <tr key={post.id}>

    {/\* Ensure to add a unique key for each row \*/}

    <td>{post.email}</td>

    <td>

      <p> {moment(post.createdAt).fromNow()}</p>

      {post.postMsg}

      <p className="likes">

        <a href="#" onClick={() => handleLikePost(post.\_id)}>

          <FaThumbsUp />

        </a>

        ({post.likes.count})

      </p>

    </td>

  </tr>

))}

</tbody>