Adding a Modal for Editing Notes | Complete R...

Show Course Contents

Overview Q&A Downloads Announcements

Adding a Modal for Editing Notes | Complete React Course in Hindi #66

In the previous article, we have added the functionality of adding and updating a Note in the database of the application with the help of API Calls. In today's article, we will be adding the functionality of deleting the note from the database of the application. In addition to this we will add a modal for editing the existing notes of the iNotebook application. So, without further ado let's begin!

Delete note API Call

In the previous article, we have made an AddNote, Fetching all Note and Updating Note API call. In a similar manner, we would be performing the deleteNote API call as shown below. The only difference is that the Delete Note endpoint URL is passed as the first parameter. Also, we are using the delete method instead of 'Get' and 'Post' as shown below:

```
// Delete a Note
const deleteNote = async (id) => {
  // API Call
  const response = await fetch(`${host}/api/notes/deletenote/${id}`, {
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
      "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
      eyJ1c2VyIjp7ImlkIjoiNjEzMWRjNWUzZTQwMzdjZDQ3MzRhMDY2In0sImlhdCI6MTYzMDY2OTU5Nn0.
      hJS0hx6I7ROugkqjL2CjrJuefA3pJi-IU5yGUbRHI4Q"
    }
  });
  const json = response.json();
  console.log(json)
```

DeleteNote Function

In NoteState.js

Delete Note Endpoint URL as the first parameter

Using the 'Delete' request as a Method

Figure 1.1: Delete Note Function

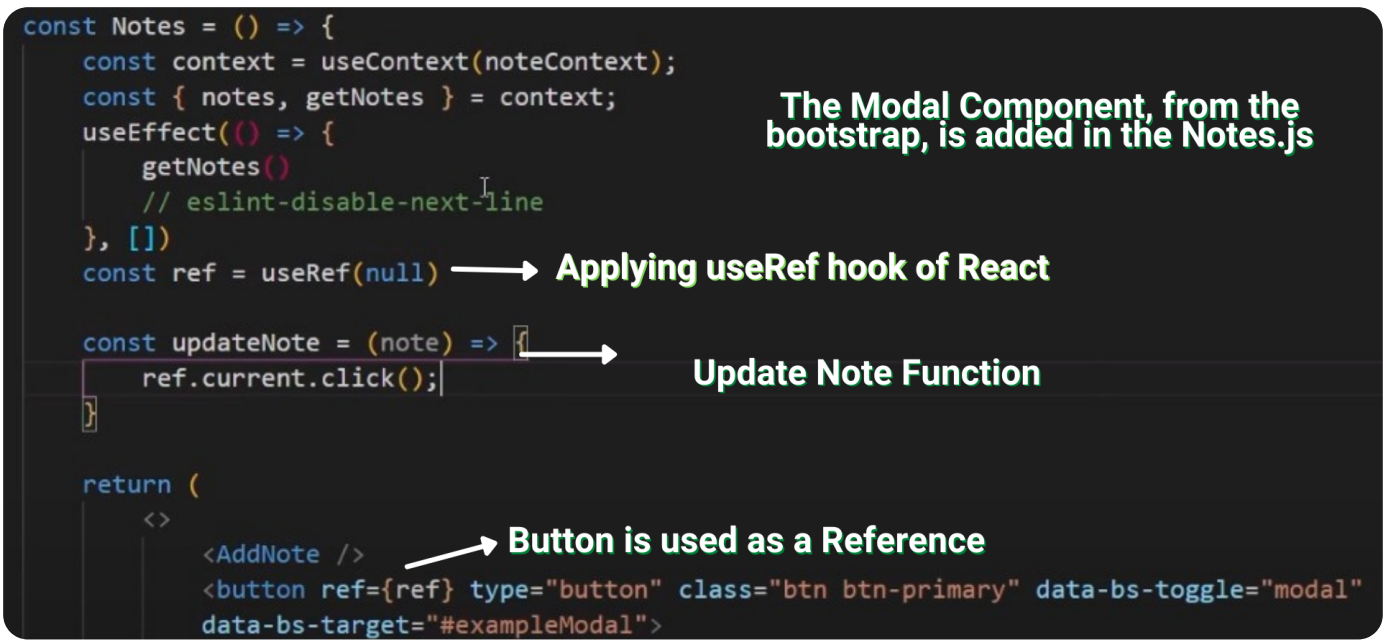
Result: Hence, we have successfully added the functionality of deleting the note from the database of the iNotebook application.

Adding a Modal For Editing Notes

Till now, we are able to fetch all the notes in the iNotebook application. In addition to this, we are able to 'Add' and 'Delete' a Note from the frontend as well as the backend of the iNotebook application. Now, our next step will be to add an 'edit a Note' functionality to the application. To do so, we will add a modal, which will be opened by clicking the edit icon. The modal will contain a form, having all the details of the specific Note, and we will be able to update the Note easily by changing the values and clicking the Update button.

Creating an Update Note function

In notes.js, First of all, we will create an Update Note function, which will take 'Note' as a parameter. We would like to invoke this function, whenever the client clicks on the update icon. Thus, this update Note function is responsible for rendering the modal of Bootstrap. Before that, let's add the code of the modal in the Notes.js by copy-pasting it from the Bootstrap as shown below:




```
const Notes = () => {
    const context = useContext(noteContext);
    const { notes, getNotes } = context;
    useEffect(() => {
        getNotes()
        // eslint-disable-next-line
    }, [])
    const ref = useRef(null)

    const updateNote = (note) => {
        ref.current.click();
    }

    return (
        <>
            <AddNote />
            <button ref={ref} type="button" class="btn btn-primary" data-bs-toggle="modal"
            data-bs-target="#exampleModal">
```

Figure 1.2: Creating an Update Note Function

Explanation: Here, we have made some desired changes in the 'Text' and 'Looks' of the Modal component of the Bootstrap. Remember, we are launching this modal with the help of Javascript. Here, we are using the 'useref' hook of React. As we have discussed earlier that it is a hook that allows us to

directly create a reference to the DOM element in the functional component. In our case, we have provided the reference to the Button of the Modal. Make sure to pass the update note function as a prop to the NoteItem and receive this function from the props, with the help of the destructuring method of javascript, in the NoteItems.js. Finally, we will assign this function to the 'onClick' event of the Edit icon, available in NoteItem.js.

Result: Hence, the modal will be launched whenever the client clicks on the Edit Icon as shown below-

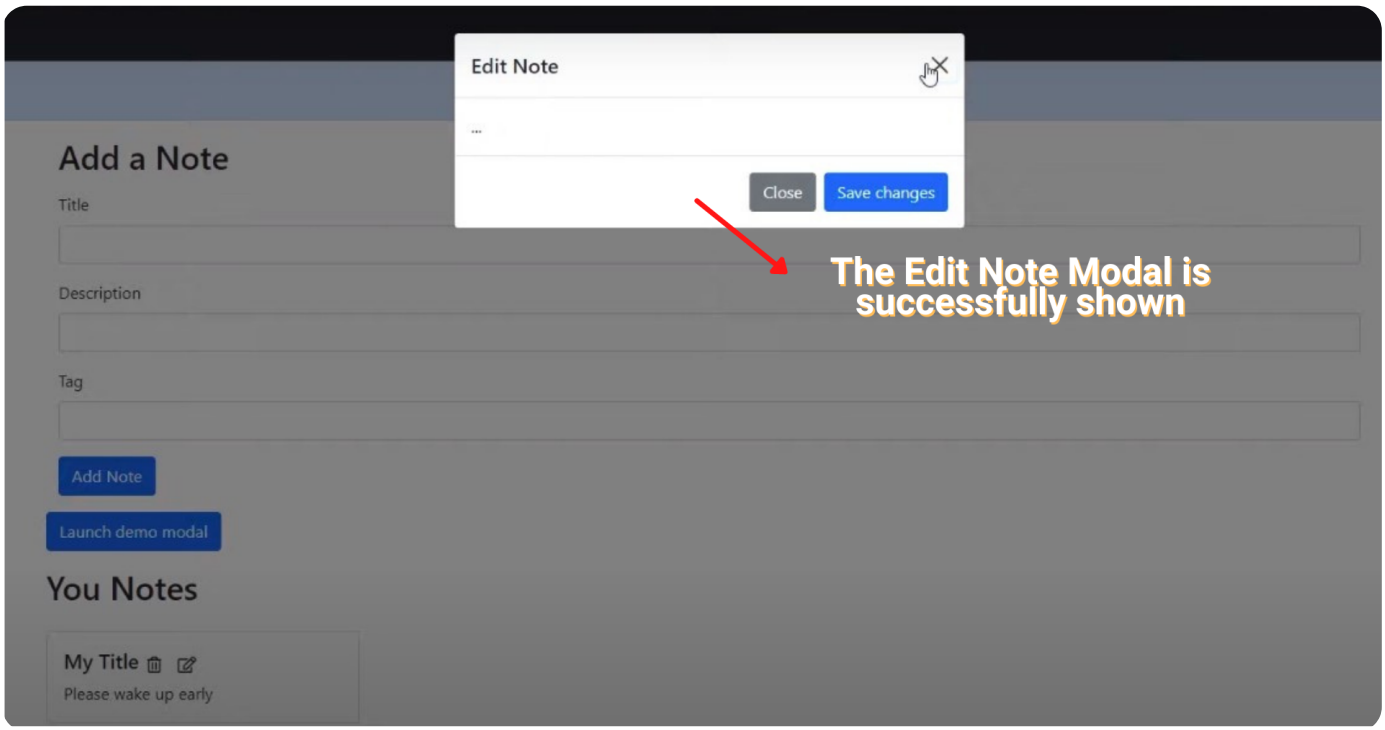


Figure 1.3: Modal - Successfully Shown

Adding the AddNote Form in the Modal

Subsequently, we would like to render the AddNote form, to edit the values of the Note, in the Edit Modal. To do so, we will copy-paste the AddNote form from the AddNote.js file. However, we have made some changes in the 'ID' and 'Name' of the fields to avoid misconceptions.

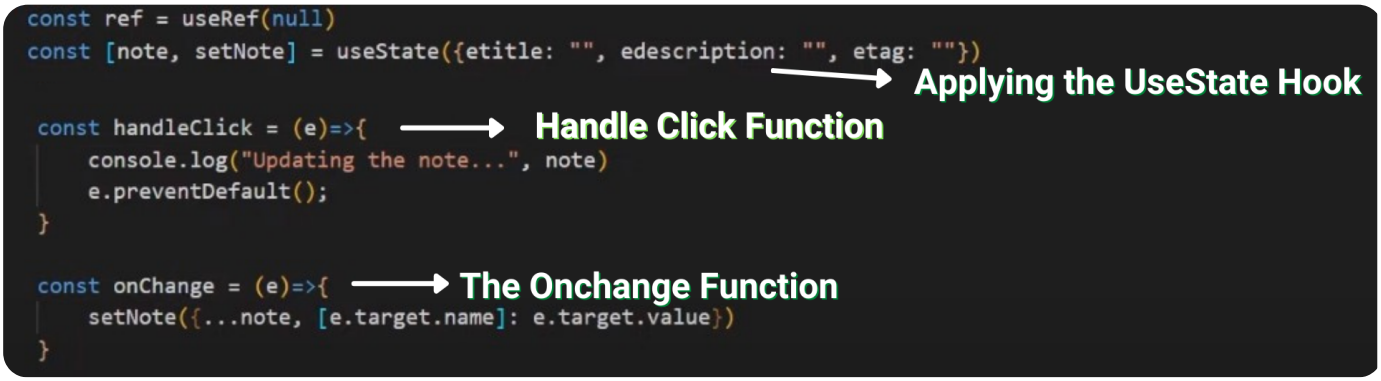


Figure 1.4: Handle Click and OnChange Function

Explanation: Above, we are adding the 'handle click' and 'Onchange' function in the Notes.js for editing the values in the fields of the form of Edit Note Modal. Here, we are also using the UseState hook to set the initial value of Title, Description, and Tag. The useState hook is a special function that takes the initial state as an argument and returns an array of two entries. If you remember, we have performed similar steps while creating the AddNote Form previously.

Result: Hence, the AddNote form is being successfully added to the Edit Note Modal as shown below -

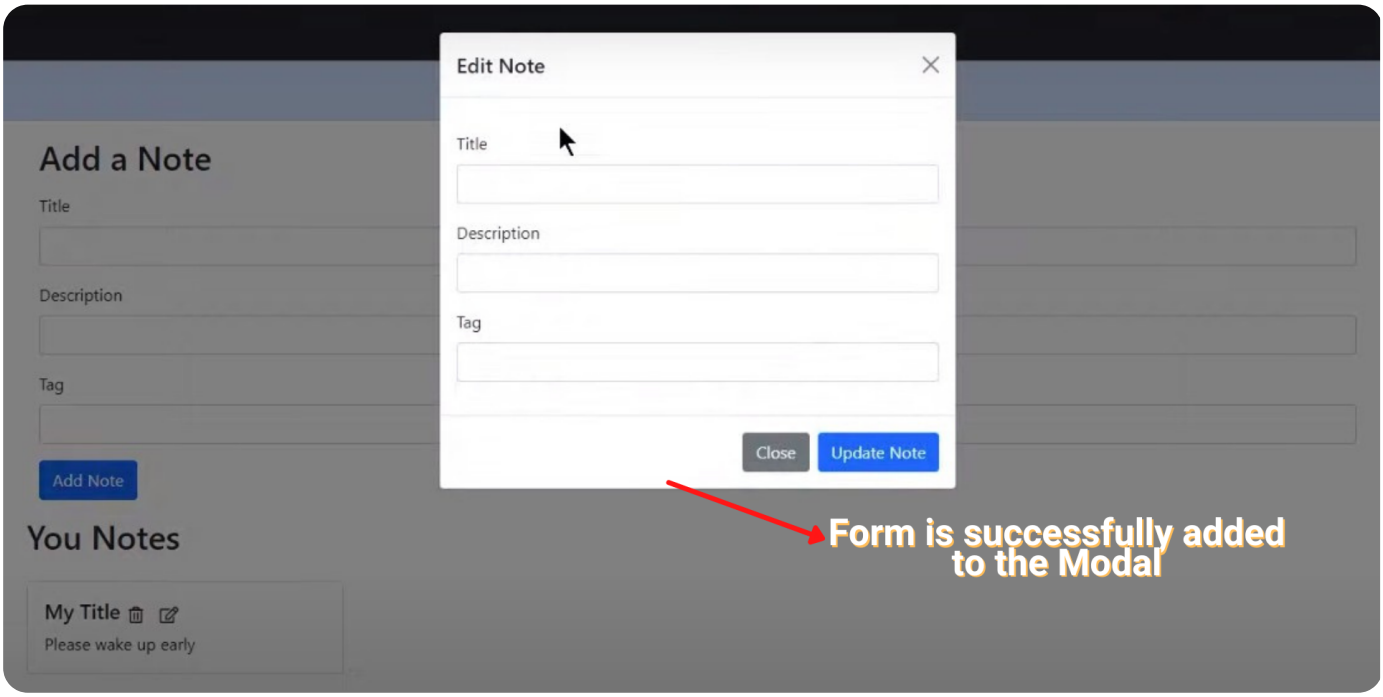


Figure 1.5: Result - Form Added in the Modal

Enhancing the Edit Note Modal Form

Now, we would like to populate the Title, Description, and Tag fields of the form with the values available in the Note. To do so, we have to populate all the values whenever the update Note Function is executed. Here's the look of the Update Note function-

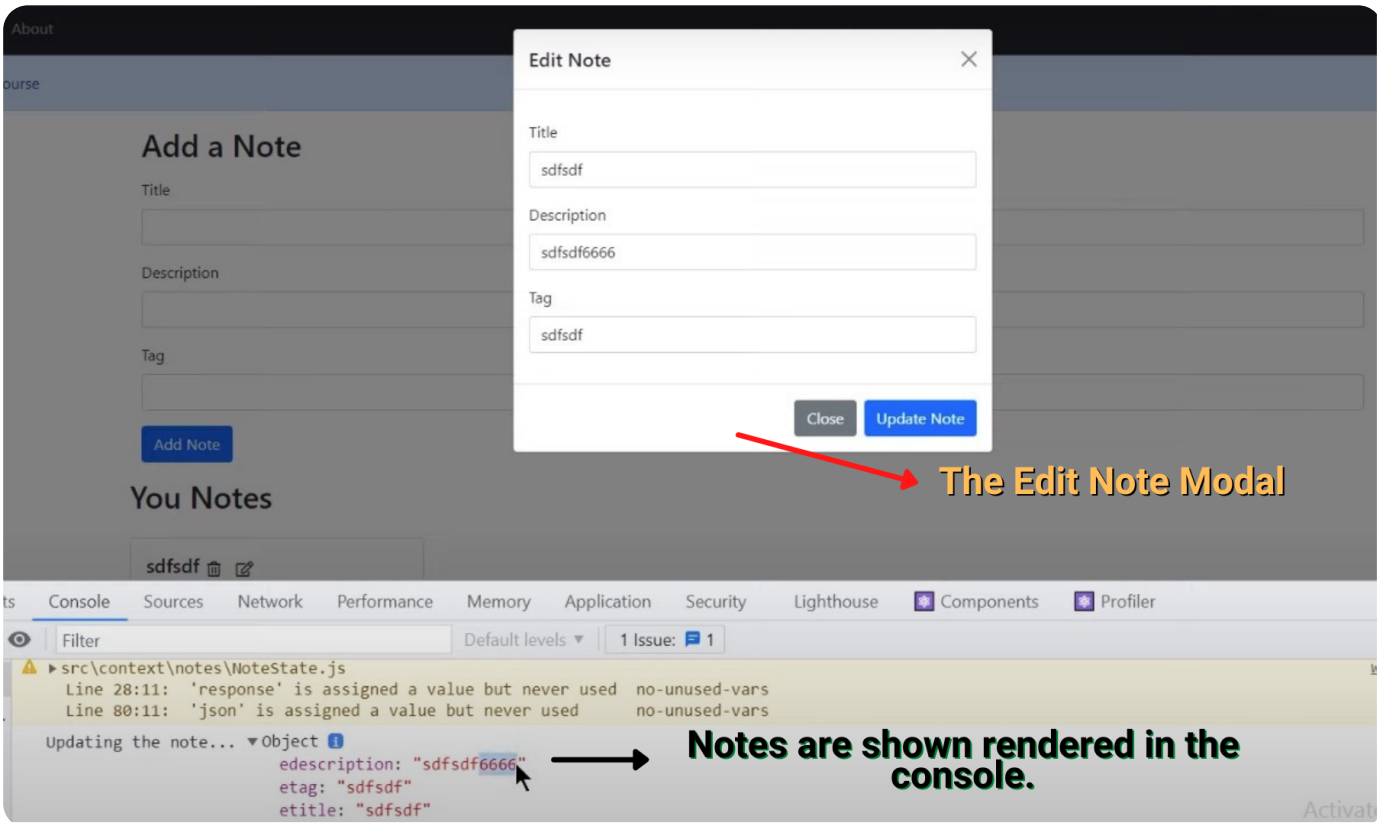
```
const ref = useRef(null)
const [note, setNote] = useState({etitle: "", edescription: "", etag: ""})

const updateNote = (currentNote) => {
    ref.current.click();
    setNote({etitle: currentNote.title, edescription: currentNote.description, etag:currentNote.tag})
}
```

Update Note function

Figure 1.6: Update Note function

Explanation: Above, we have enhanced the Update Note function. Here, we have passed the current Note as a parameter in the Update Note function and have used the setNote() method. Thus, we have set the values of the 'current note' in the Fields of the Edit Modal Form. Make sure to assign the values to all the fields of the Form, available in Notes.js. Also, Assign the Handle Click function to the Onclick event of the Update Note button of the Edit Note Form.

Result: Now, the edit note modal contains the forms, which have the existing values of the title, description and Tag fields of the Note. Moreover, the 'onchange' is also working efficiently, as we are able to change the values of the title, description and tag fields of the Form. Most importantly, on clicking the update note button, the handle Click function is being successfully executed. Therefore, the updated Notes are rendered as an object in the Console. However, the Notes aren't updated in the UI of the iNotebook application.



some errors in our application and will be understanding some amazing concepts of React.

Do remember to Access the playlist of the Free React Course in Hindi by clicking here.

Thank you for being with me throughout the tutorial. In the next tutorial we will be Updating Notes on edit in the UI using React. Till then, Keep learning and Keep Coding.

Previous Next

CodeWithHarry