User Dashboard Creation for Blog's

1. **Create a functionality or button to display Post Creation form where User/Author can fill the title and content to create the post.**

   Button to show the Post Creation Form

   Form to create a new post (title + content)

   Display all posts created by the user

   Edit an existing post

   Soft delete a post (mark as deleted, but not remove from database)

2. **To display all the Post create by the user.**

1. React (Frontend) – `UserDashboard.jsx`

```
import React, { useState, useEffect } from "react";


function UserDashboard() {
  const [posts, setPosts] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [formData, setFormData] = useState({ title: "", content: "", id: null
});


  useEffect(() => {
    fetch("/api/posts/my")
      .then(res => res.json())
      .then(data => setPosts(data));
  }, []);


  const handleFormSubmit = async (e) => {
    e.preventDefault();
    const method = formData.id ? "PUT" : "POST";
```

```
    const endpoint = formData.id ? `/api/posts/${formData.id}` :
"/api/posts";

    await fetch(endpoint, {
      method,
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ title: formData.title, content: formData.content
}),
    });

    resetForm();
    loadPosts();
  };

  const resetForm = () => {
    setFormData({ title: "", content: "", id: null });
    setShowForm(false);
  };

  const loadPosts = async () => {
    const response = await fetch("/api/posts/my");
    const data = await response.json();
    setPosts(data);
  };

  const editPost = (post) => {
    setFormData({ title: post.title, content: post.content, id: post._id });
    setShowForm(true);
  };

  const deletePost = async (id) => {
```

```
    await fetch(`/api/posts/${id}`, { method: "DELETE" });
    loadPosts();
  };

  return (
    <div>
      <h2>User Dashboard</h2>
      <button onClick={() => setShowForm(!showForm)}>
        {showForm ? "Cancel" : "Create Post"}
      </button>

      {showForm && (
        <form onSubmit={handleFormSubmit}>
          <input
            type="text"
            placeholder="Title"
            value={formData.title}
            onChange={(e) => setFormData({ ...formData, title: e.target.value
})}
            required
          />
          <textarea
            placeholder="Content"
            value={formData.content}
            onChange={(e) => setFormData({ ...formData, content:
e.target.value })}
            required
          ></textarea>
          <button type="submit">{formData.id ? "Update Post" : "Create
Post"}</button>
        </form>
```

```
      )}

      <ul>
        {posts.map(post => (
          <li key={post._id}>
            <h3>{post.title}</h3>
            <p>{post.content}</p>
            <button onClick={() => editPost(post)}>Edit</button>
            <button onClick={() => deletePost(post._id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default UserDashboard;
```

3. Functionality to Edit the Post

Backend (Node.js + Express)

`models/Post.js`

```
const mongoose = require('mongoose');

const PostSchema = new mongoose.Schema({
  title: String,
  content: String,
  userId: mongoose.Schema.Types.ObjectId,
  deleted: { type: Boolean, default: false },
}, { timestamps: true });

module.exports = mongoose.model('Post', PostSchema);
```

routes/posts.js

```
const express = require('express');
const router = express.Router();
const Post = require('../models/Post');

// Simulated user ID (replace with real auth logic)
const currentUserId = "663e9eab45778c8b25123abc";

// Get user's posts (not deleted)
router.get('/my', async (req, res) => {
  const posts = await Post.find({ userId: currentUserId, deleted: false });
  res.json(posts);
});

// Create post
router.post('/', async (req, res) => {
  const { title, content } = req.body;
  const post = new Post({ title, content, userId: currentUserId });
  await post.save();
  res.status(201).json(post);
});

// Edit post
router.put('/:id', async (req, res) => {
  const { title, content } = req.body;
  const updated = await Post.findOneAndUpdate(
    { _id: req.params.id, userId: currentUserId },
    { title, content },
    { new: true }
  );
```

```
  res.json(updated);
});

// Soft delete
router.delete('/:id', async (req, res) => {
  await Post.findOneAndUpdate(
    { _id: req.params.id, userId: currentUserId },
    { deleted: true }
  );
  res.sendStatus(204);
});

module.exports = router;
app.js
const express = require('express');
const mongoose = require('mongoose');
const postRoutes = require('./routes/posts');
const app = express();

mongoose.connect("mongodb://localhost:27017/blogdb");

app.use(express.json());
app.use('/api/posts', postRoutes);

app.listen(3001, () => console.log("API listening on port 3001"));
```

4. **Functionality to Delete the Post [Soft delete]**

   Instead of removing the document, we set `deleted: true` and exclude such posts from the display queries.