

## deleteButton.jsx

```
import { deleteDoc, doc } from "firebase/firestore";
import { db } from "../firebase";

export default function DeleteButton({ postId }) {
  const handleDelete = async () => {
    const docRef = doc(db, "posts", postId);
    try {
      await deleteDoc(docRef);
      window.location.reload();
    } catch (error) {
      console.error("削除できませんでした", error);
    }
  };

  return <button onClick={handleDelete}>削除</button>;
}
```

## Likebutton.jsx

```
import { useCallback, useEffect, useState } from "react";
import { auth, db } from "../firebase";
import {
  collection,
  deleteDoc,
  doc,
  onSnapshot,
  setDoc,
} from "firebase/firestore";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faHeart } from "@fortawesome/free-solid-svg-icons";

export function LikeButton({ postId }) {
  const userId = auth.currentUser?.uid;

  const [isLiked, setIsLiked] = useState(null);

  const [likeCount, setLikeCount] = useState(0);
```

```

useEffect(() => {
  if (!userId) return;

  const postRef = doc(db, "posts", postId);
  const likedUserRef = doc(postRef, "LikedUsers", userId);

  const unsubscribeLikedUser = onSnapshot(likedUserRef, (doc) => {
    setIsLiked(doc.exists());
  });

  const likedUsersRef = collection(db, `posts/${postId}/LikedUsers`);

  const unsubscribeLikedCount = onSnapshot(likedUsersRef, (snapshot) => {
    setLikeCount(snapshot.size);
  });

  return () => {
    unsubscribeLikedUser();
    unsubscribeLikedCount();
  };
}, [userId, postId]);

const handleClick = useCallback(async () => {
  if (!userId || isLiked === null) return;

  const likedUserRef = doc(db, "posts", postId, "LikedUsers", userId);

  const userLikePostRef = doc(db, "users", userId, "likePosts", postId);
  if (isLiked) {
    await deleteDoc(likedUserRef);
    await deleteDoc(userLikePostRef);
  } else {
    await setDoc(likedUserRef, { userId });
    await setDoc(userLikePostRef, { postId });
  }
}, [userId, postId, isLiked]);

if (!userId) return null;
if (isLiked === null) return null;

return (
  <div>

```

```

    <button onClick={handleClick}>
      <FontAwesomeIcon icon={faHeart} color={isLiked ? "red" : "grey"} />
      <p>{likeCount}</p>
    </button>
  </div>
);
}

```

## Login.jsx

```

import { useCallback, useEffect, useState } from "react";
import { auth, db } from "../firebase";
import {
  collection,
  deleteDoc,
  doc,
  onSnapshot,
  setDoc,
} from "firebase/firestore";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faHeart } from "@fortawesome/free-solid-svg-icons";

export function LikeButton({ postId }) {
  const userId = auth.currentUser?.uid;

  const [isLiked, setIsLiked] = useState(null);

  const [likeCount, setLikeCount] = useState(0);

  useEffect(() => {
    if (!userId) return;

    const postRef = doc(db, "posts", postId);
    const likedUserRef = doc(postRef, "LikedUsers", userId);

    const unsubscribeLikedUser = onSnapshot(likedUserRef, (doc) => {
      setIsLiked(doc.exists());
    });

    const likedUsersRef = collection(db, `posts/${postId}/LikedUsers`);
  });
}

```

```

const unsubscribeLikedCount = onSnapshot(likedUsersRef, (snapshot) => {
  setLikeCount(snapshot.size);
});

return () => {
  unsubscribeLikedUser();
  unsubscribeLikedCount();
};
}, [userId, postId]);

const handleClick = useCallback(async () => {
  if (!userId || isLiked === null) return;

  const likedUserRef = doc(db, "posts", postId, "LikedUsers", userId);

  const userLikePostRef = doc(db, "users", userId, "likePosts", postId);
  if (isLiked) {
    await deleteDoc(likedUserRef);
    await deleteDoc(userLikePostRef);
  } else {
    await setDoc(likedUserRef, { userId });
    await setDoc(userLikePostRef, { postId });
  }
}, [userId, postId, isLiked]);

if (!userId) return null;
if (isLiked === null) return null;

return (
  <div>
    <button onClick={handleClick}>
      <FontAwesomeIcon icon={faHeart} color={isLiked ? "red" : "grey"} />
      <p>{likeCount}</p>
    </button>
  </div>
);
}

```

## Index.jsx

```
import { Link } from "@remix-run/react";
import Login from "../components/Login";
import "../styles/design.css";

export const meta = () => {
  return [
    { title: "New Remix SPA" },
    { name: "description", content: "Welcome to Remix (SPA Mode)!" },
  ];
};

export default function Index() {
  return (
    <div
      className="LoginPage"
      style={{ fontFamily: "system-ui, sans-serif", lineHeight: "1.8" }}
    >
      <header className="LoginPage-header">
        <h1>高校生向け匿名サイト</h1>
        <Login />
        <ul>
          <li>
            <Link to="/createPost">投稿画面へ</Link>
          </li>
          <li>
            <Link to="/postIndex">投稿一覧へ</Link>
          </li>
        </ul>
      </header>
    </div>
  );
}
```

## createPost.jsx

```
import { collection, addDoc } from "firebase/firestore";
import { useState } from "react";
import { db } from "../firebase";
import { useNavigate } from "react-router-dom";
import Select from "react-select";
```

```
import { Link } from "@remix-run/react";

export default function CreatePost() {
  const [title, setTitle] = useState("");
  const [content, setContent] = useState("");
  const [selectedCategory, setSelectedCategory] = useState(null);
  const navigate = useNavigate();

  const category = [
    { value: "kousoku", label: "#校則" },
    { value: "bukatsu", label: "#部活" },
    { value: "shinro", label: "#進路" },
    { value: "kadai", label: "#課題" },
  ];

  const handleSubmit = async (e) => {
    e.preventDefault();

    await addDoc(collection(db, "posts"), {
      title: title,
      content: content,
      category: selectedCategory ? selectedCategory.label : "カテゴリなし",
    });

    setTitle("");
    setContent("");
    setSelectedCategory(null);

    navigate("/postIndex");
  };

  return (
    <header className="CreatePostPage-header">
      <h1>新規投稿</h1>
      <form onSubmit={handleSubmit}>
        <label htmlFor="title">タイトル:</label>
        <input
          id="title"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
          style={{
            display: "block",

```

```

padding: "10px",
margin: "10px 0",
width: "80%",
maxWidth: "400px",
}}
placeholder="内容:ex) 校則"
/>

<label htmlFor="content">本文:</label>
<textarea
  id="content"
  value={content}
  onChange={(e) => setContent(e.target.value)}
/>

<label htmlFor="category">カテゴリ:</label>
<Select
  id="category"
  options={category}
  value={selectedCategory}
  onChange={setSelectedCategory}
/>

<button type="submit">投稿する</button>
</form>

<Link to="/postIndex" className="page-change">
  投稿一覧
</Link>
<aside className="related-posts">
  <h3>関連投稿</h3>
  <ul>
    <li>関連投稿1</li>
    <li>関連投稿2</li>
    <li>関連投稿3</li>
  </ul>
</aside>
</header>
);
}

```

## postIndex.jsx

```
import { collection, getDocs } from "firebase/firestore";
import { useState, useEffect } from "react";
import { db } from "../firebase";
import { LikeButton } from "../components/likebutton";
import { Link } from "@remix-run/react";
import DeleteButton from "../components/deleteButton";

export default function GetPosts() {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    const fetchPosts = async () => {
      const querySnapshot = await getDocs(collection(db, "posts"));
      const postData = querySnapshot.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      setPosts(postData);
    };

    fetchPosts();
  }, []);

  return (
    <div className="PostListPage">
      <header className="CreatePostPage-header">
        <h1>投稿一覧</h1>
        <Link to="/createPost" className="page-change">
          新規投稿
        </Link>
      </header>
      <ul className="post-list">
        {posts.map((post) => (
          <li key={post.id}>
            <h3>{post.title}</h3>
            <p>{post.content}</p>
            <p>{post.category || "カテゴリなし"}</p>
            {post.id && <LikeButton postId={post.id} />}
          </li>
        ))}
      </ul>
    </div>
  );
}
```



```

        {post.id && <DeleteButton postId={post.id} />}
      </li>
    )}}
  </ul>
  <aside className="related-posts">
    <h3>関連投稿</h3>
    <ul>
      <li>関連投稿1</li>
      <li>関連投稿2</li>
      <li>関連投稿3</li>
    </ul>
  </aside>
</div>
);
}

```

## Design.css

```

.LoginPage,
.CreatePostPage,
.PostListPage {
  text-align: center;
  font-family: Arial, sans-serif;
  padding: 20px;
}

.LoginPage-header,
.CreatePostPage-header,
.PostListPage-header {
  background-color: #282c34;
  padding: 20px;
  color: white;
}

input,
textarea {
  padding: 10px;
  margin-bottom: 10px;
  display: block;
  width: 80%;
}

```

```
margin-left: auto;
margin-right: auto;
}

button {
padding: 10px 20px;
background-color: #8cd5ea;
border: none;
color: white;
cursor: pointer;
}

button:hover {
background-color: #ef89bb;
}

form {
margin: 20px 0;
}

.related-posts {
text-align: left;
padding: 20px;
}

.post-list {
display: flex;
flex-direction: column;
align-items: center;
}

.page-change {
text-align: right;
font-family: Arial, sans-serif;
padding: 20px;
}
```