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>;
}
```

lkebutton.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>
        <section id="instructions">
          <h2>投稿の手順</h2>
          <ol>
            <li>投稿画面へ移動します。</li>
            <li>タイトルと内容を入力します。</li>
            <li>「投稿」ボタンをクリックします。</li>
            <li>投稿が表示されます。</li>
          </ol>
        </section>
      </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 postsData = querySnapshot.docs.map((doc) => ({
       id: doc.id,
       ...doc.data(),
     }));
     setPosts(postsData);
   };

   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} />}
            {post.id && <DeleteButton postId={post.id} />}
          </li>
        ))}
      </ul>
      <aside className="related-posts">
        <h3>関連投稿</h3>
        <ul>
          <section id="instructions">
            <h2>投稿の手順</h2>
            <ol>
              <li>投稿画面へ移動します。</li>
              <li>タイトルと内容を入力します。</li>
              <li>「投稿」ボタンをクリックします。</li>
              <li>投稿が表示されます。</li>
            </ol>
          </section>
        </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;
}

body {
 font-family: Arial, sans-serif;
 background-color: #f0f8ff;
 color: #333;
 margin: 0;
 padding: 0;
}

header {
 background-color: #4682b4;
 color: white;
 padding: 20px;
 text-align: center;
}

header h1 {
 margin: 0;
 font-size: 2.5em;
}

nav ul {
 list-style-type: none;
 padding: 0;
}

nav ul li {
 display: inline;
 margin: 0 10px;
}

nav ul li a {
 color: white;
 text-decoration: none;
 font-weight: bold;
}

main {
```

```
 padding: 20px;
}

#instructions {
 background-color: #e0f7fa;
 padding: 15px;
 border-radius: 10px;
 margin-bottom: 20px;
}

ol {
 padding-left: 20px;
}

#ranking {
 background-color: #ffe0b2;
 padding: 15px;
 border-radius: 10px;
}

button {
 background-color: #ff5722;
 color: white;
 border: none;
 padding: 10px 20px;
 border-radius: 5px;
 cursor: pointer;
 transition: background-color 0.3s ease;
}

button:hover {
 background-color: #e64a19;
}

.like-button {
 display: flex;
 align-items: center;
 cursor: pointer;
}

.like-button p {
 margin: 0 0 0 5px;
```

```
 font-size: 1.2em;
 transition: color 0.3s ease;
}

.like-button:hover p {
 color: red;
}
```