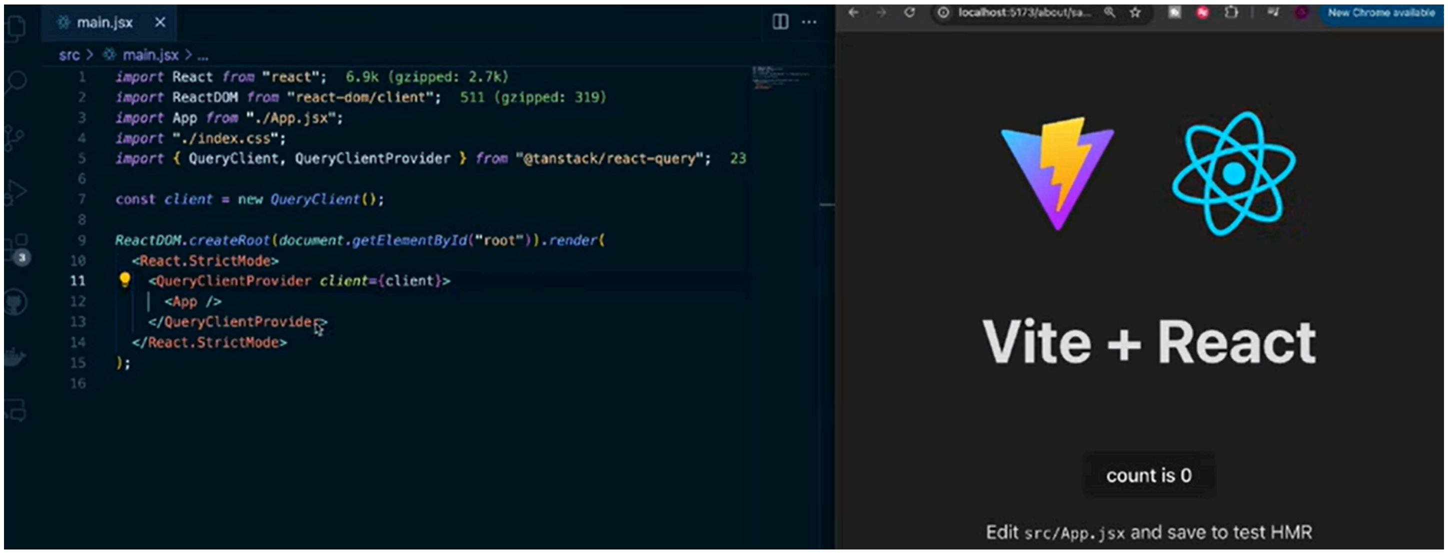
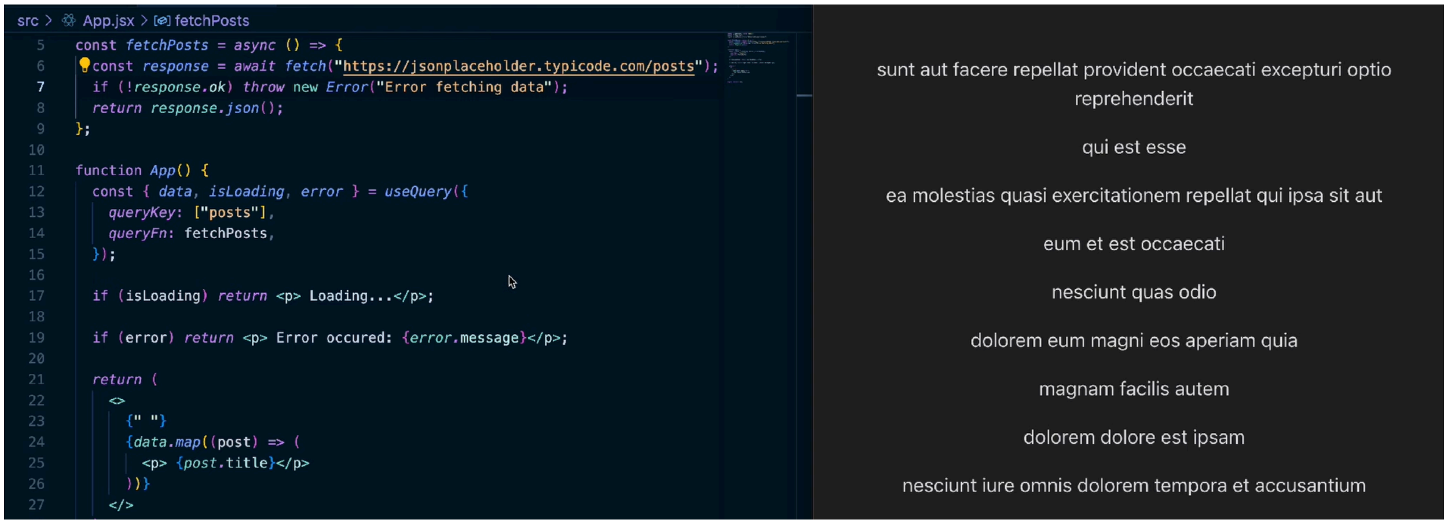
## Step 1 for TanStack



```
import React from "react";  6.9k (gzipped: 2.7k)
import ReactDOM from "react-dom/client";  511 (gzipped: 319)
import App from "./App.jsx";
import "./index.css";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";  23

const client = new QueryClient();

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <QueryClientProvider client={client}>
      <App />
    </QueryClientProvider>
  </React.StrictMode>
);
```

## Fetch Data



```
const fetchPosts = async () => {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts");
  if (!response.ok) throw new Error("Error fetching data");
  return response.json();
};

function App() {
  const { data, isLoading, error } = useQuery({
    queryKey: ["posts"],
    queryFn: fetchPosts,
  });

  if (isLoading) return <p> Loading...</p>;

  if (error) return <p> Error occured: {error.message}</p>;

  return (
    <>
      {" "}
      {data.map((post) => (
        <p> {post.title}</p>
      ))}
    </>
```

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

qui est esse

ea molestias quasi exercitationem repellat qui ipsa sit aut

eum et est occaecati

nesciunt quas odio

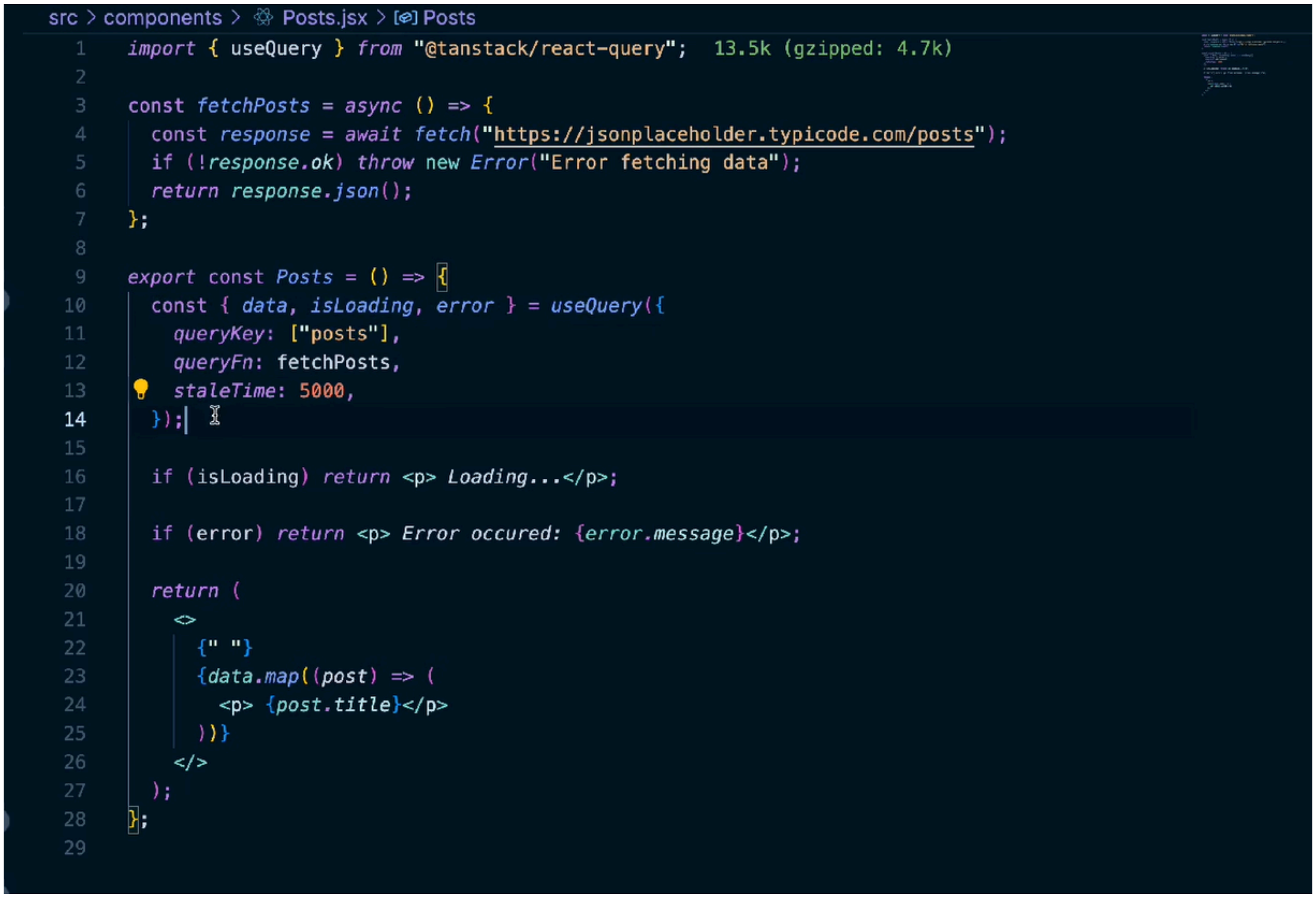
dolorem eum magni eos aperiam quia

magnam facilis autem

dolorem dolore est ipsam

nesciunt iure omnis dolorem tempora et accusantium

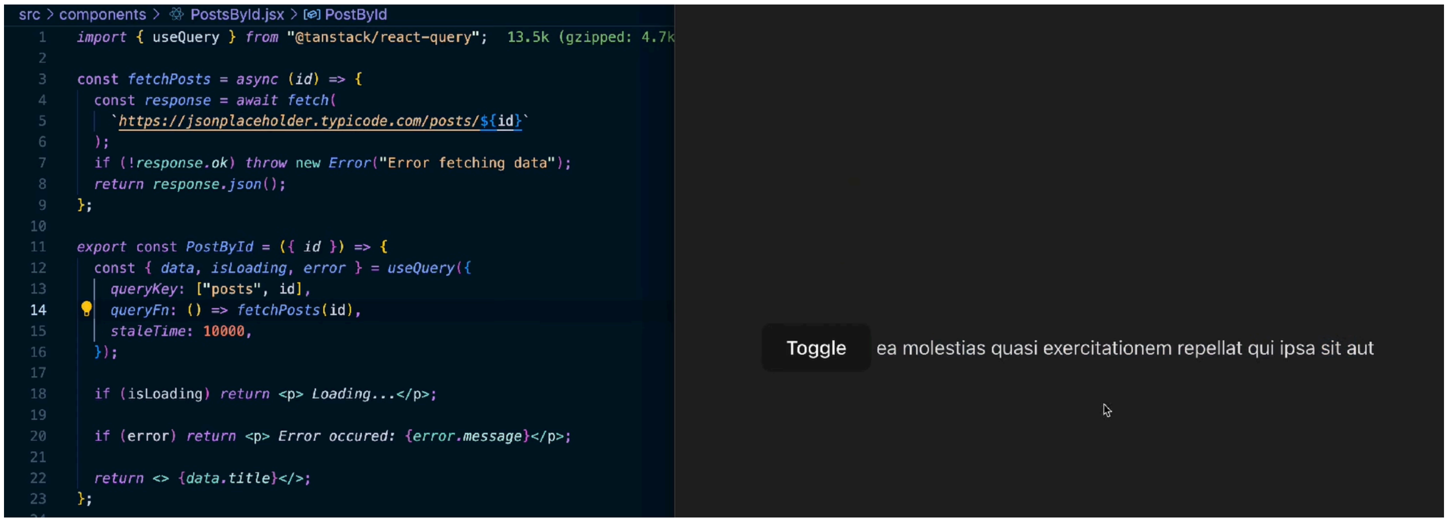
Keep fetching data after a certain time period (In this example, data is fetched every 5 seconds (5000ms))



src > components > Posts.jsx > Posts

```jsx
import { useQuery } from "@tanstack/react-query";  13.5k (gzipped: 4.7k)

const fetchPosts = async () => {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts");
  if (!response.ok) throw new Error("Error fetching data");
  return response.json();
};

export const Posts = () => {
  const { data, isLoading, error } = useQuery({
    queryKey: ["posts"],
    queryFn: fetchPosts,
    staleTime: 5000,
  });

  if (isLoading) return <p> Loading...</p>;

  if (error) return <p> Error occured: {error.message}</p>;

  return (
    <>
      {" "}
      {data.map((post) => (
        <p> {post.title}</p>
      ))}
    </>
  );
};
```

Fetch data using ID

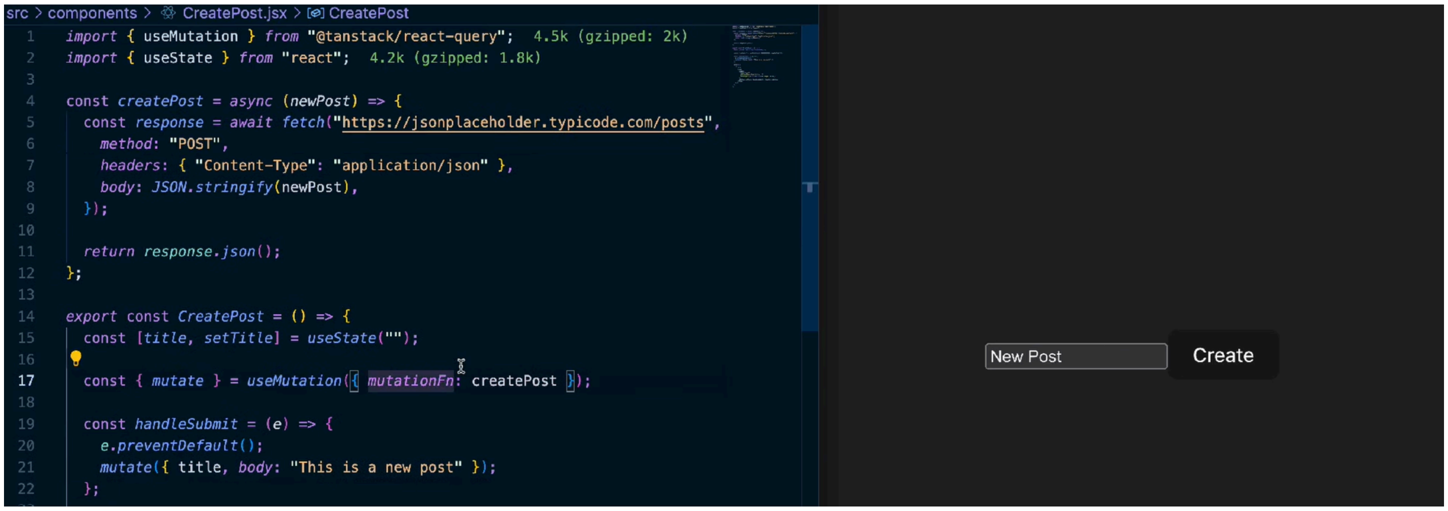


src > components > PostsById.jsx > PostById

```jsx
import { useQuery } from "@tanstack/react-query";  13.5k (gzipped: 4.7k

const fetchPosts = async (id) => {
  const response = await fetch(
    `https://jsonplaceholder.typicode.com/posts/${id}`
  );
  if (!response.ok) throw new Error("Error fetching data");
  return response.json();
};

export const PostById = ({ id }) => {
  const { data, isLoading, error } = useQuery({
    queryKey: ["posts", id],
    queryFn: () => fetchPosts(id),
    staleTime: 10000,
  });

  if (isLoading) return <p> Loading...</p>;

  if (error) return <p> Error occured: {error.message}</p>;

  return <> {data.title}</>;
};
```

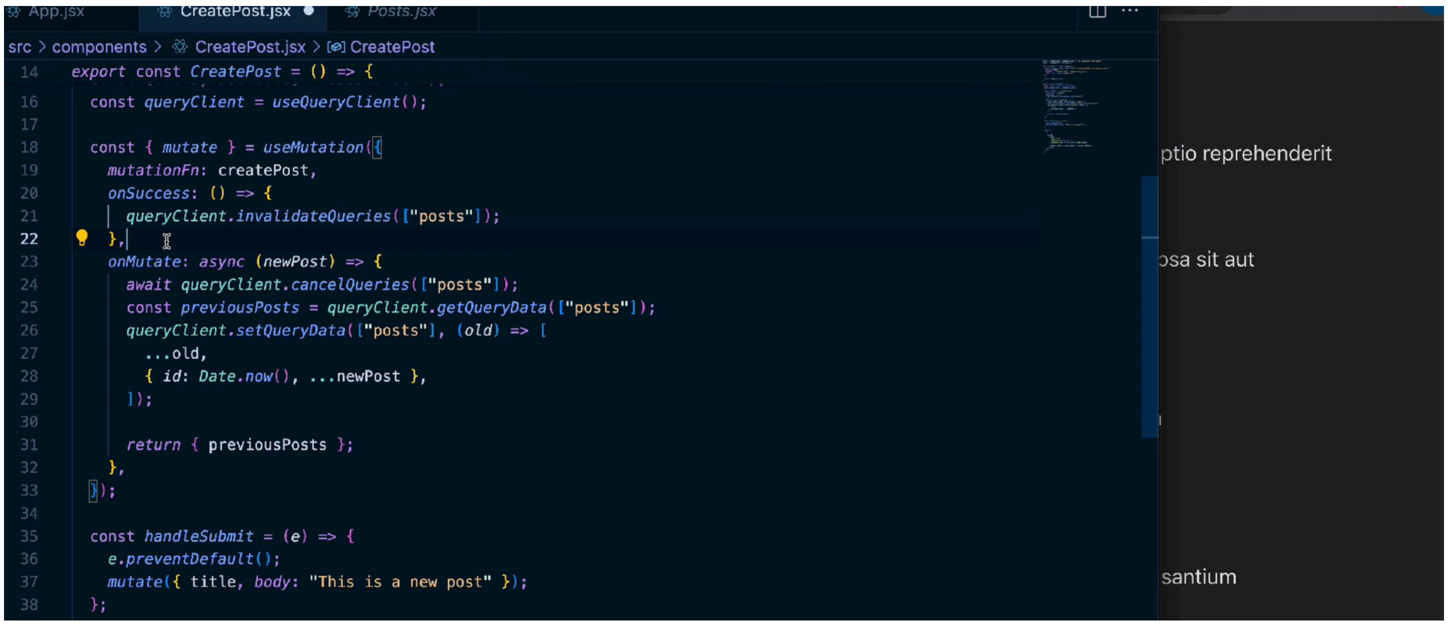
ea molestias quasi exercitationem repellat qui ipsa sit aut

Posting data


src > components > CreatePost.jsx > CreatePost

```
import { useMutation } from "@tanstack/react-query";  4.5k (gzipped: 2k)
import { useState } from "react";  4.2k (gzipped: 1.8k)

const createPost = async (newPost) => {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts",
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(newPost),
  });

  return response.json();
};

export const CreatePost = () => {
  const [title, setTitle] = useState("");

  const { mutate } = useMutation({ mutationFn: createPost });

  const handleSubmit = (e) => {
    e.preventDefault();
    mutate({ title, body: "This is a new post" });
  };
```

Implementing optimistic updates



src > components > CreatePost.jsx > CreatePost

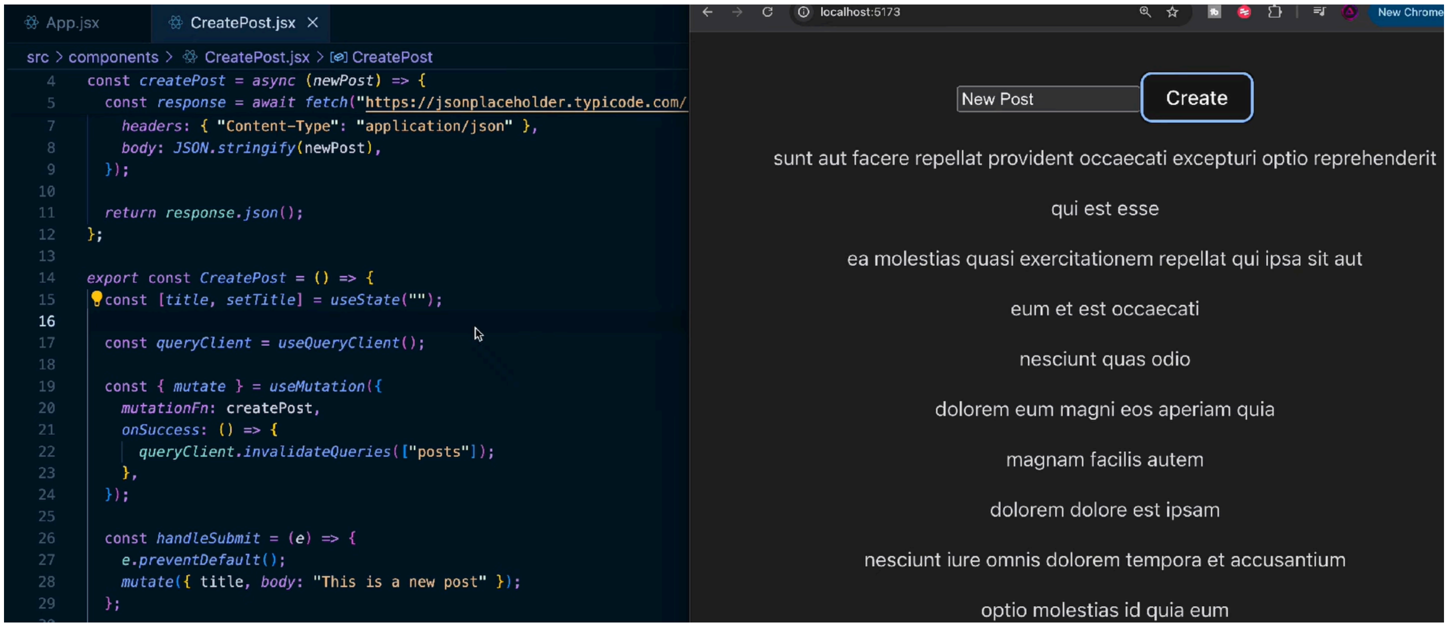
```
export const CreatePost = () => {
  const queryClient = useQueryClient();

  const { mutate } = useMutation({
    mutationFn: createPost,
    onSuccess: () => {
      queryClient.invalidateQueries(["posts"]);
    },
    onMutate: async (newPost) => {
      await queryClient.cancelQueries(["posts"]);
      const previousPosts = queryClient.getQueryData(["posts"]);
      queryClient.setQueryData(["posts"], (old) => [
        ...old,
        { id: Date.now(), ...newPost },
      ]);

      return { previousPosts };
    },
  });

  const handleSubmit = (e) => {
    e.preventDefault();
    mutate({ title, body: "This is a new post" });
  };
```

ptio reprehenderit

osa sit aut

santium

Referring to other queries



```
const createPost = async (newPost) => {
  const response = await fetch("https://jsonplaceholder.typicode.com/
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(newPost),
  });

  return response.json();
};

export const CreatePost = () => {
  const [title, setTitle] = useState("");

  const queryClient = useQueryClient();

  const { mutate } = useMutation({
    mutationFn: createPost,
    onSuccess: () => {
      queryClient.invalidateQueries(["posts"]);
    },
  });

  const handleSubmit = (e) => {
    e.preventDefault();
    mutate({ title, body: "This is a new post" });
  };
```

New Post
Create

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

qui est esse

ea molestias quasi exercitationem repellat qui ipsa sit aut

eum et est occaecati

nesciunt quas odio

dolorem eum magni eos aperiam quia

magnam facilis autem

dolorem dolore est ipsam

nesciunt iure omnis dolorem tempora et accusantium

optio molestias id quia eum