**Exercise 23: Lazy Components**

Objectives and Outcomes

This exercise is a demo that showcases how to use lazy components. In this example, you can create lazy loading components for a user and post feature that fetches data from an API, using React 18 and React Suspense.

Exercises

An example of lazy loading components for a user and post feature that fetches data from an API, using React 18 and React Suspense

* Create two components, User and Post, in separate files:

// User.js

import React from 'react';

const User = ({ user }) => {

return (

<div>

<h2>{user.name}</h2>

<p>{user.email}</p>

</div>

);

};

export default User;

// Post.js

import React from 'react';

const Post = ({ post }) => {

return (

<div>

<h2>{post.title}</h2>

<p>{post.body}</p>

</div>

);

};

export default Post;

* Create an API client module to fetch user and post data:

// api.js

export const fetchUser = async (userId) => {

const response = await fetch(`https://jsonplaceholder.typicode.com/users/${userId}`);

const user = await response.json();

return user;

};

export const fetchPost = async (postId) => {

const response = await fetch(`https://jsonplaceholder.typicode.com/posts/${postId}`);

const post = await response.json();

return post;

};

* Create a component that will handle the lazy loading and rendering of the components:

// App.js

import React, { Suspense } from 'react';

import { fetchUser, fetchPost } from './api';

const User = React.lazy(() => import('./User'));

const Post = React.lazy(() => import('./Post'));

const App = () => {

return (

<div>

<h1>Lazy Loading Demo</h1>

<Suspense fallback={<div>Loading...</div>}>

<User user={fetchUser(1)} />

<Post post={fetchPost(1)} />

</Suspense>

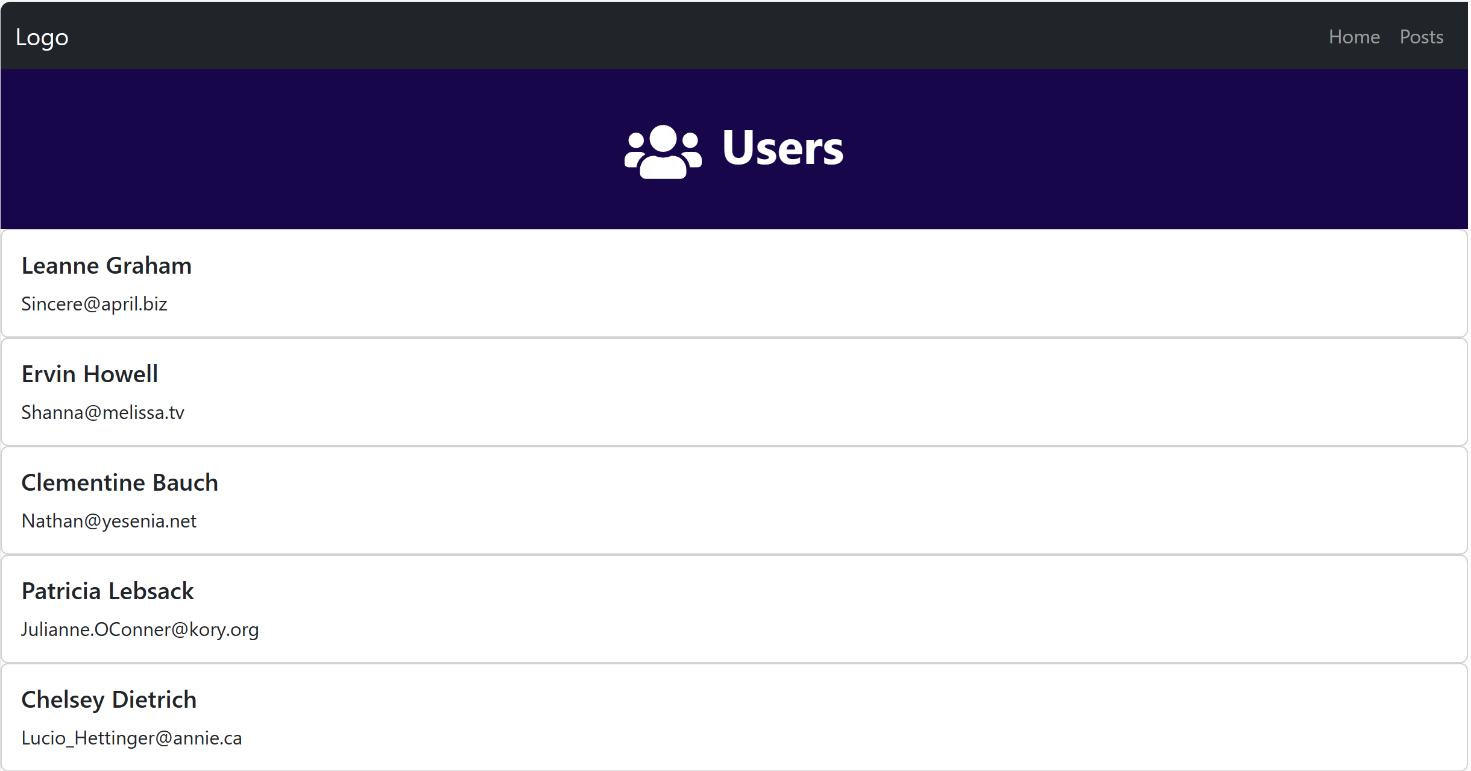
</div>

);

};

export default App;

From an example abow, list all users and posts with lazy components





Conclusion

Lazy loading combined with data fetching from an API can help optimize the performance of your React application by loading components and data on demand.