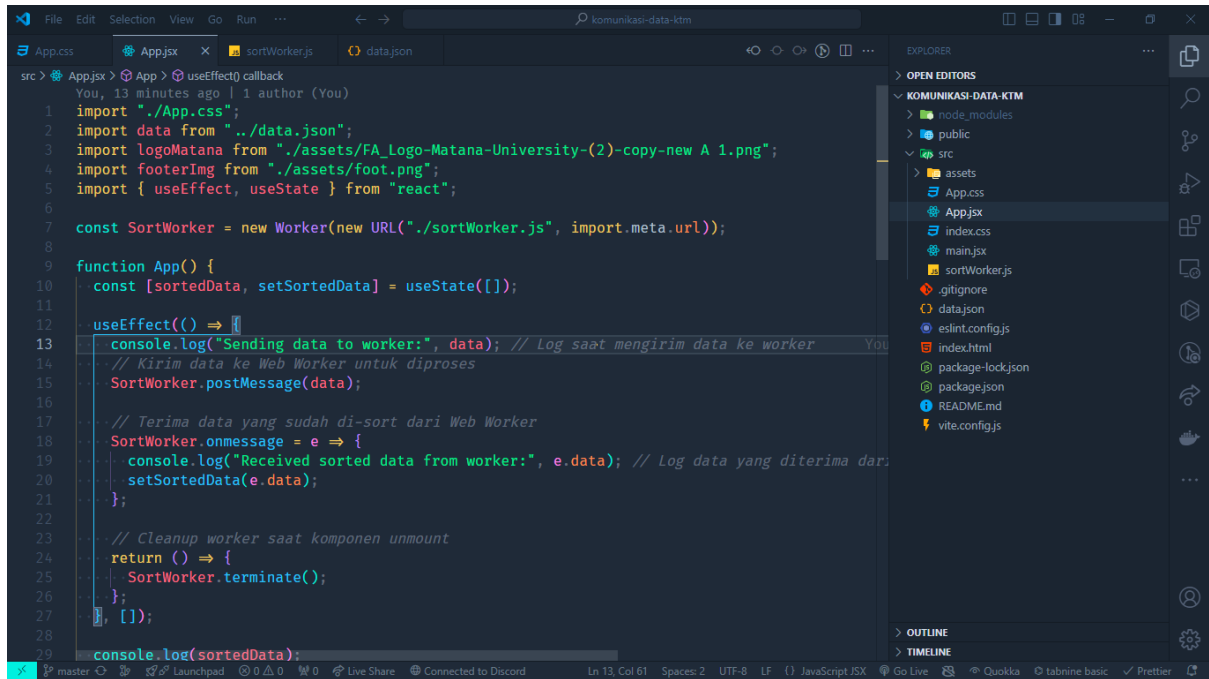


UAS Komunikasi Data

Penilaian Individu, Jeriko Ichtus Seo(20235520002)

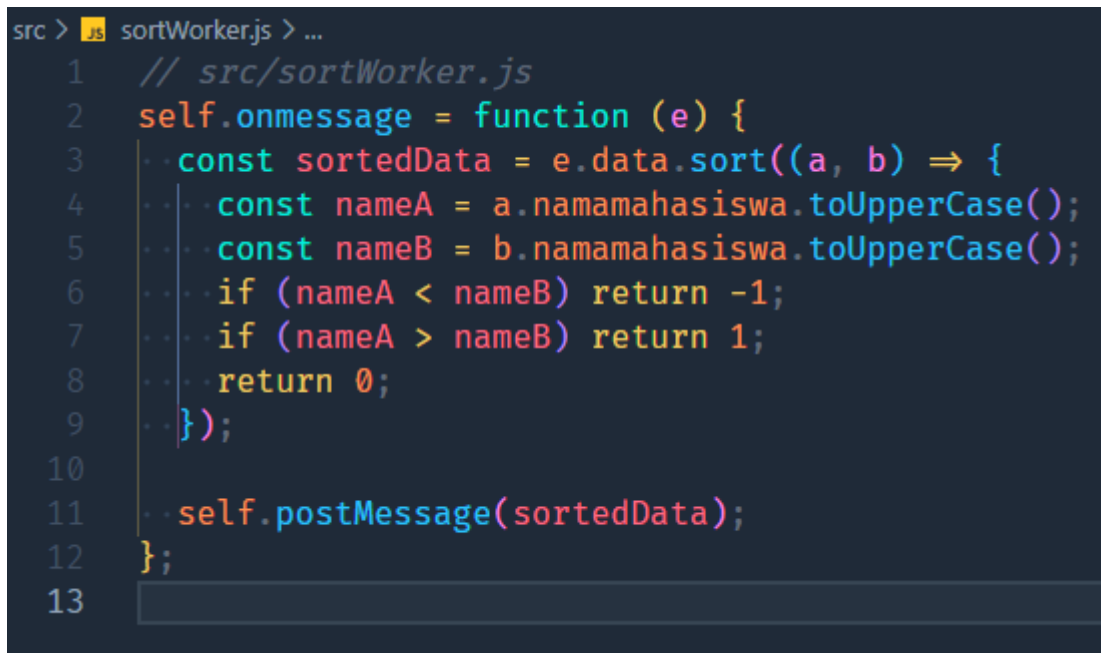
App.jsx



```
src > App.jsx > App > useEffect() callback
You, 13 minutes ago | 1 author (You)
1 import "../App.css";
2 import data from "../data.json";
3 import logoMatana from "../assets/FA_Logo-Matana-University-(2)-copy-new A 1.png";
4 import footerIng from "../assets/foot.png";
5 import { useEffect, useState } from "react";
6
7 const SortWorker = new Worker(new URL("./sortWorker.js", import.meta.url));
8
9 function App() {
10   const [sortedData, setSortedData] = useState([]);
11
12   useEffect(() => {
13     console.log("Sending data to worker:", data); // Log saat mengirim data ke worker
14     // Kirim data ke Web Worker untuk diproses
15     SortWorker.postMessage(data);
16
17     // Terima data yang sudah di-sort dari Web Worker
18     SortWorker.onmessage = e => {
19       console.log("Received sorted data from worker:", e.data); // Log data yang diterima dari
20       setSortedData(e.data);
21     };
22
23     // Cleanup worker saat komponen unmount
24     return () => {
25       SortWorker.terminate();
26     };
27   }, []);
28
29   // console.log(sortedData);

```

selfWorker.js



```
src > sortWorker.js > ...
1 // src/sortWorker.js
2 self.onmessage = function (e) {
3   const sortedData = e.data.sort((a, b) => {
4     const nameA = a.namamahasiswa.toUpperCase();
5     const nameB = b.namamahasiswa.toUpperCase();
6     if (nameA < nameB) return -1;
7     if (nameA > nameB) return 1;
8     return 0;
9   });
10
11   self.postMessage(sortedData);
12 }
13
```

Penjelasan kode file selfWorker.js

Kode yang berada di **selfWorker.js** menggunakan Web Worker untuk mengurutkan data nama mahasiswa secara alfabet tanpa membebani thread utama.

1. self.onmessage = function(e):

- Mendengarkan pesan dari main thread.
- Data yang dikirim ada di e.data.

2. const sortedData = e.data.sort(...):

- Mengurutkan array yang berisi objek mahasiswa berdasarkan properti namamahasiswa.
- Nama mahasiswa diubah jadi huruf besar agar pengurutan tidak sensitif huruf besar/kecil.

3. Pengurutan:

- Jika nama a lebih kecil dari nama b, a ditempatkan lebih dulu.
- Jika nama a lebih besar, b ditempatkan lebih dulu.
- Jika nama sama, posisi tidak berubah.

4. self.postMessage(sortedData):

- Setelah diurutkan, hasilnya dikirim kembali ke main thread.

Tujuannya adalah mengurutkan data mahasiswa tanpa membuat UI lambat.

Penjelasan kode file App.jsx

Kode di atas adalah sebuah **React component** bernama Kode di atas adalah sebuah **React component** bernama App yang menampilkan data mahasiswa secara dinamis setelah diurutkan menggunakan **Web Worker**. yang menampilkan data mahasiswa secara dinamis setelah diurutkan menggunakan **Web Worker**.

Imports

- **import './App.css';**: Mengimpor file CSS untuk styling komponen.
- **import data from './data.json';**: Mengimpor data mahasiswa dari file data.json.
- **import logoMatana from './assets/FA_Logo-Matana-University-(2)-copy-new A 1.png';**: Mengimpor gambar logo Matana University.

- **import footerImg from "../assets/foot.png";**: Mengimpor gambar untuk bagian footer.
- **import { useEffect, useState } from "react";**: Mengimpor hooks `useEffect` dan `useState` dari React.

2. Web Worker Initialization

- **const SortWorker = new Worker(new URL("../sortWorker.js", import.meta.url));**: Membuat instance dari Web Worker dengan mengacu ke file `sortWorker.js` yang akan menangani pengurutan data di thread terpisah.

3. React Component App

- **const [sortedData, setSortedData] = useState([]);**: Menggunakan state `sortedData` untuk menyimpan data mahasiswa yang sudah diurutkan. Inisialisasi awal adalah array kosong.

4. useEffect Hook

- **useEffect(() => { ... }, []);**: Hook ini dijalankan setelah komponen di-render untuk pertama kali. Isinya mengirim data ke Web Worker untuk diurutkan dan menerima data yang sudah diurutkan dari worker.
- **SortWorker.postMessage(data);**: Mengirim data mahasiswa dari file `data.json` ke Web Worker untuk diolah.
- **SortWorker.onmessage = e => { ... };**: Menangani pesan yang diterima dari worker setelah proses pengurutan selesai. Hasilnya adalah data yang sudah diurutkan yang disimpan di state `sortedData`.
 - **setSortedData(e.data);**: Memperbarui state `sortedData` dengan data mahasiswa yang telah diurutkan.
- **return () => { SortWorker.terminate(); };**: Mengakhiri worker ketika komponen di-unmount (tidak lagi ditampilkan) untuk menghindari kebocoran memori.

5. Render Output

- **{sortedData.length > 0 ? (...) : (<p>Loading data...</p>)}**: Kondisi yang digunakan untuk menampilkan data mahasiswa yang sudah diurutkan. Jika `sortedData` sudah ada (jumlahnya lebih dari 0), data tersebut akan ditampilkan. Jika tidak, akan menampilkan teks "Loading data..." sebagai placeholder.

6. Rendering Data Mahasiswa

- **sortedData.map((d, i) => (...))**: Data mahasiswa diulang menggunakan `map()`. Untuk setiap item `d` (data mahasiswa), sebuah elemen `div` dibuat untuk menampilkan informasi mahasiswa.

- ****: Menampilkan gambar foto mahasiswa. Jika d.foto tidak ada, digunakan gambar default "src.jpg".
- **<h3 className="mahasiswa">{d.namamahasiswa}</h3>**: Menampilkan nama mahasiswa.
- **<h3 className="prodi">{d.programstudi}</h3>**: Menampilkan program studi mahasiswa.
- **<h4 className="nim">student code : {d.nim}</h4>**: Menampilkan NIM mahasiswa.
- **Footer Image**: Gambar footer ditampilkan di setiap elemen div mahasiswa menggunakan **footerImg**.

7. Penggunaan Gambar

- **logoMatana**: Logo universitas ditampilkan di bagian atas setiap item mahasiswa.
- **footerImg**: Gambar di bagian bawah setiap item mahasiswa.