LAPORAN FINAL PROJECT PEMROGRAMAN TERINTEGRASI

Disusun oleh :

Shafira Faiz Aulia Winanda	22082010044
Muhammad Rafli Alfarisqi	22082010213
Alfatur Rabbani	22082010194

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UPN "VETERAN JAWA TIMUR"

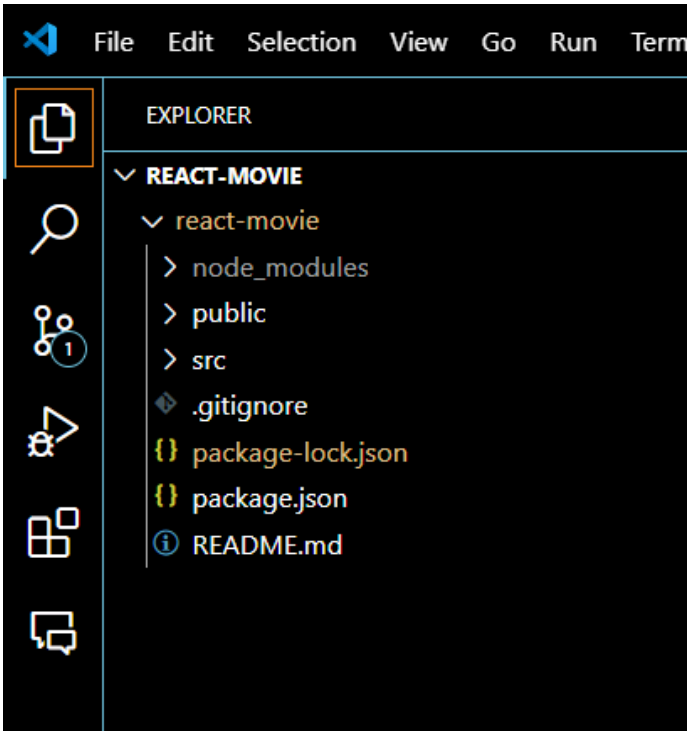
2024

A. Deksripsi Proyek

Proyek React Movie RMDB adalah aplikasi web berbasis ReactJS yang dirancang untuk menyediakan database film yang cepat, responsif, dan mudah digunakan. Aplikasi ini memanfaatkan The Movie DB API untuk menampilkan informasi terkini mengenai berbagai film populer, termasuk sinopsis, rating, poster, dan detail aktor. Dengan fitur pencarian yang intuitif, navigasi ke halaman detail film, serta tampilan yang menarik dan responsif, aplikasi ini memudahkan pengguna dalam menjelajahi kategori film dan memperoleh informasi secara langsung. Proyek ini juga menjadi referensi yang ideal bagi pengembang yang ingin membangun aplikasi React dengan integrasi API eksternal untuk pengelolaan data secara dinamis.

B. Struktur kode

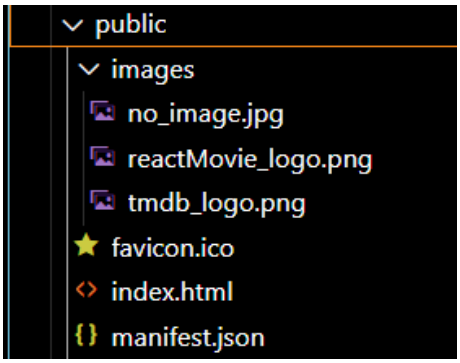
Proyek the movie yang kami buat memiliki beberapa kerangka dasar aplikasi yang dibuat menggunakan alat seperti Create React App.



Struktur folder utama aplikasi meliputi:

1. **Folder node_modules,** menyimpan semua dependensi atau library yang diperlukan oleh aplikasi.

2. **Folder public,** menyimpan file statis seperti gambar dan file HTML. Berikut isi detailnya :



- images (Folder):
  Folder ini berisi file gambar yang digunakan dalam aplikasi. Dalam folder ini terdapat:
  - no_image.jpg: Gambar placeholder yang digunakan jika tidak ada gambar lain yang tersedia.
  - reactMovie_logo.png: Logo aplikasi React Movie.
  - tmdb_logo.png: Logo TMDB (The Movie Database), kemungkinan digunakan untuk mencantumkan sumber data.
- favicon.ico (File):
  Ikon kecil yang muncul di tab browser saat aplikasi dijalankan. Favicon ini menjadi identitas visual aplikasi.
- index.html (File):

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <meta name="theme-color" content="#000000" />
    <!--
      manifest.json provides metadata used when your web app is added to the
      homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" />

    <link
      href="https://fonts.googleapis.com/css?family=Abel"
      rel="stylesheet"
    />
    <link
      href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
      rel="stylesheet"
    />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React Movie</title>
  </head>
  <body>
    <noscript> You need to enable JavaScript to run this app. </noscript>
```

```
  </head>
  <body>
    <noscript> You need to enable JavaScript to run this app. </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

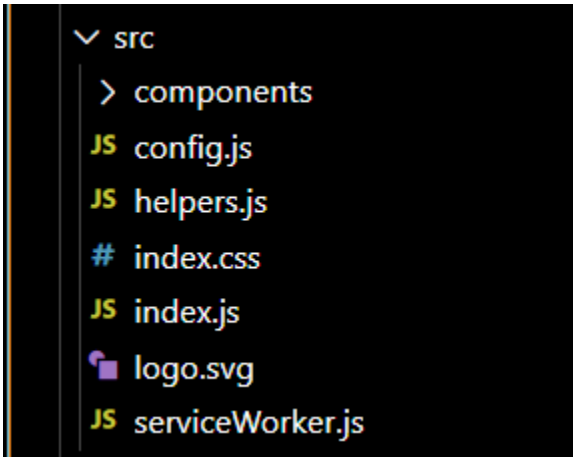
File HTML utama yang menjadi titik awal aplikasi React. React akan merender aplikasinya ke dalam elemen di dalam file ini, biasanya dengan ID root.

- manifest.json (File):

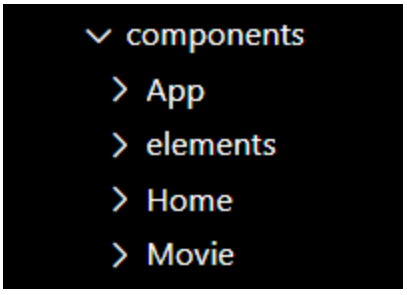
```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    }
  ],
  "start_url": "./index.html",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

File konfigurasi untuk Progressive Web App (PWA). File ini berisi metadata seperti nama aplikasi, ikon, dan warna tema yang digunakan saat aplikasi ditambahkan ke layar utama perangkat pengguna.

Folder src memiliki beberapa struktur yang terdiri dari :



- components (Folder):



Folder ini berisi komponen-komponen React yang membangun antarmuka pengguna. Komponen-komponen ini dibagi ke dalam beberapa subfolder, antara lain:

- App

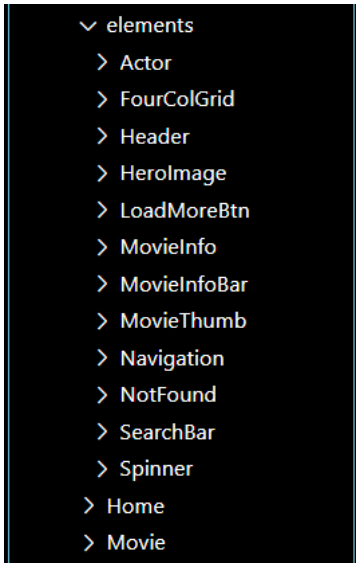
```
import React from "react";
import { BrowserRouter, Route, Switch } from "react-router-dom";
import Header from "../elements/Header/Header";
import Home from "../Home/Home";
import Movie from "../Movie/Movie";
import NotFound from "../elements/NotFound/NotFound";

class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <React.Fragment>
          <Header />
          <Switch>
            <Route path="/" component={Home} exact />
            <Route path="/:movieId" component={Movie} exact />
            <Route component={NotFound} />
          </Switch>
        </React.Fragment>
      </BrowserRouter>
    );
  }
}

export default App;
```

Kode diatas merupaka app.js isi dari folder app yang berisi komponen-komponen React yang membangun antarmuka pengguna. Komponen-komponen ini dibagi ke dalam beberapa subfolder, antara lain:

- Element



Folder ini berisi kumpulan komponen yang reusable (dapat digunakan ulang) dalam aplikasi. Komponen-komponen tersebut merepresentasikan bagian-bagian kecil dari UI yang sering digunakan. Komponen di dalamnya meliputi:

- Actor :
  - Actor.css: Gaya untuk komponen Actor.
  - Actor.js: Komponen yang menampilkan aktor dalam film.
- FourColGrid:
  - FourColGrid.css: Gaya untuk tata letak grid.
  - FourColGrid.js: Komponen untuk menampilkan daftar film atau aktor dalam grid empat kolom.
- Header:
  - Header.css: Gaya untuk header.
  - Header.js: Komponen untuk menampilkan header aplikasi.
- HeroImage:
  - HeroImage.css: Gaya untuk gambar utama.
  - HeroImage.js: Komponen yang menampilkan gambar besar di halaman awal.
- LoadMoreBtn:
  - LoadMoreBtn.css: Gaya untuk tombol "Load More".
  - LoadMoreBtn.js: Komponen untuk menampilkan tombol untuk memuat lebih banyak konten.
- MovieInfo:
  - MovieInfo.css: Gaya untuk informasi film.
  - MovieInfo.js: Komponen untuk menampilkan detail film.
- MovieInfoBar:
  - MovieInfoBar.css: Gaya untuk bar informasi film.
  - MovieInfoBar.js: Komponen untuk menampilkan informasi tambahan tentang film.
- MovieThumb:
  - MovieThumb.css: Gaya untuk thumbnail film.
  - MovieThumb.js: Komponen untuk menampilkan gambar thumbnail film.
- Navigation:
  - Navigation.css: Gaya untuk navigasi.
  - Navigation.js: Komponen untuk menampilkan breadcrumb navigasi.

- NotFound:
    - NotFound.js: Komponen untuk menangani halaman yang tidak ditemukan.
- SearchBar :
    - SearchBar.css: Gaya untuk bilah pencarian.
    - SearchBar.js: Komponen untuk pencarian film.
- Spinner:
    - Spinner.css: Gaya untuk animasi loading.
    - Spinner.js: Komponen untuk menampilkan animasi loading.
- Home
    - Home.css : Gaya untuk halaman beranda.
    - Home.js : Komponen beranda yang merangkum elemen utama aplikasi.
- Movie
    - Movie.css : Gaya untuk halaman detail film.
    - Movie.js : Komponen untuk menampilkan detail sebuah film tertentu.
- Config.js

```
// Configuration for TMDB
// To se the latest configuration fetch it from https://api.themoviedb.org/3/configuration?api_key=019e8f375549e0bbd4a4191862ebc88f

const API_URL = 'https://api.themoviedb.org/3/';
const API_KEY = '844dba0bfd8f3a4f3799f6130ef9e335';

// Images
// An image URL looks like this example:
// http://image.tmdb.org/t/p/w780/bOGkgRGdhrBYJSLpXaxhXVstddV.jpg

const IMAGE_BASE_URL ='http://image.tmdb.org/t/p/';

//Sizes: w300, w780, w1280, original
const BACKDROP_SIZE = 'w1280';

// w92, w154, w185, w342, w500, w780, original
const POSTER_SIZE = 'w500';

export {
  API_URL,
  API_KEY,
  IMAGE_BASE_URL,
  BACKDROP_SIZE,
  POSTER_SIZE
}
```

Kode di atas adalah file konfigurasi untuk aplikasi yang terhubung dengan TMDB API. Kode ini mendefinisikan URL dasar API (API_URL), kunci API (API_KEY) untuk autentikasi, dan URL dasar untuk mengambil gambar (IMAGE_BASE_URL). Selain itu, kode ini menentukan ukuran gambar yang digunakan, yaitu backdrop dengan resolusi 1280 piksel (BACKDROP_SIZE) dan poster dengan resolusi 500 piksel (POSTER_SIZE). Semua variabel ini diekspor agar dapat digunakan di file lain dalam aplikasi, sehingga mempermudah pengelolaan koneksi API dan mengurangi duplikasi kode. File

ini berfungsi sebagai pusat konfigurasi, sehingga setiap perubahan terkait API hanya perlu dilakukan di satu tempat.

- Helper.js

```
// Convert time to hours and minutes
export const calcTime = (time) => {
  const hours = Math.floor(time / 60);
  const mins = time % 60;
  return `${hours}h ${mins}m`;
}

// Convert a number to $ format
export const convertMoney = (money) => {
  var formatter = new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD',
    minimumFractionDigits: 0,
  });
  return formatter.format(money);
}
```

Kode di atas berisi dua fungsi utilitas untuk manipulasi data. Fungsi pertama, calcTime, digunakan untuk mengonversi waktu dalam satuan menit menjadi format jam dan menit. Fungsi ini membagi waktu dengan 60 untuk menghitung jam, menggunakan sisa pembagian untuk menit, lalu mengembalikan hasil dalam format seperti "2h 30m". Fungsi kedua, convertMoney, digunakan untuk mengonversi angka menjadi format mata uang USD. Dengan memanfaatkan Intl.NumberFormat, fungsi ini memformat angka dalam gaya mata uang dengan simbol dolar tanpa desimal dan mengembalikan hasil yang sudah diformat. Kedua fungsi ini sering digunakan untuk menampilkan data waktu dan uang dengan format yang lebih mudah dibaca oleh pengguna.

- Index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./components/App/App";
import "./index.css";

ReactDOM.render(<App />, document.getElementById("root"));

```

Kode di atas merupakan entry point utama untuk aplikasi React. Baris-baris pertama mengimpor modul dan file yang diperlukan, termasuk library React, ReactDOM, komponen utama aplikasi (App), dan file CSS untuk styling. Pada baris terakhir, fungsi ReactDOM.render() digunakan untuk merender komponen App ke dalam elemen HTML dengan ID root, yang biasanya terdapat di file HTML utama aplikasi. Dengan cara ini, seluruh aplikasi React diinisialisasi dan diintegrasikan ke dalam DOM pada elemen root, memungkinkan interaksi aplikasi dengan browser.

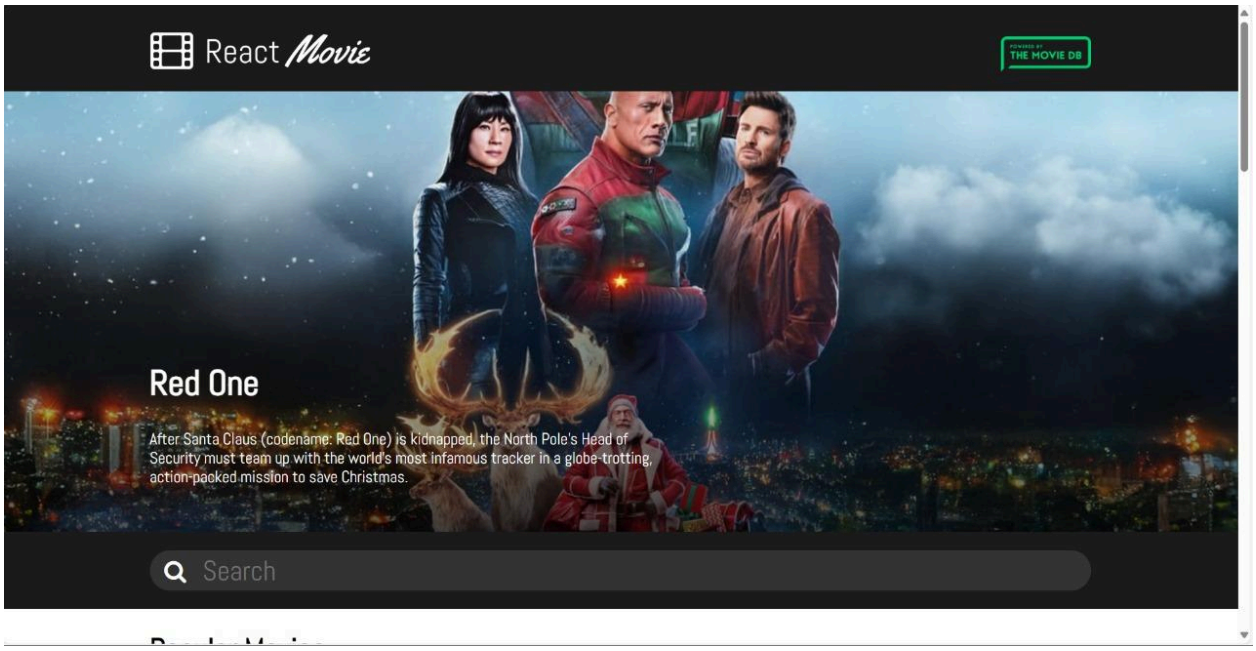
- Package.json

Kode di atas adalah file package.json, yang berfungsi sebagai metadata dan pengelola dependensi untuk proyek React. File ini mendefinisikan nama proyek (react_movie_db_course), versinya, serta daftar dependensi yang digunakan, seperti react, react-dom, dan react-router-dom, yang diperlukan untuk menjalankan aplikasi. Selain itu, terdapat bagian scripts yang menyediakan perintah untuk menjalankan berbagai tugas, seperti memulai aplikasi secara lokal (start), membangun aplikasi untuk produksi (build), menjalankan pengujian (test), dan menghapus konfigurasi build default (eject). File ini merupakan inti dari pengelolaan proyek Node.js dan memastikan semua dependensi yang diperlukan dapat diinstal dan dijalankan dengan benar.

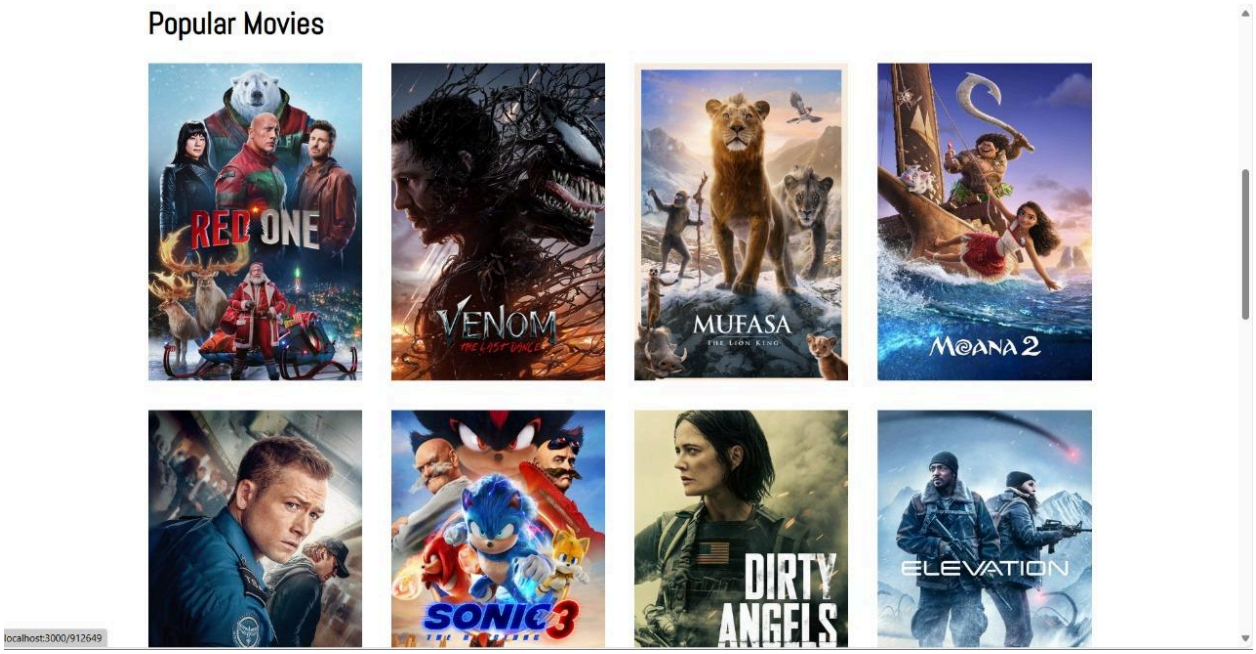
C. Langkah-langkah membuat website API

1. Pastikan Node.js sudah terinstal.
2. Inisialisasi proyek dengan npx create-react-app the-movie.
3. Tambahkan dependensi seperti Axios untuk komunikasi API dan React Router untuk navigasi.
4. Buat file config.js untuk menyimpan URL endpoint API dan kunci API.
5. Gunakan Axios untuk mengambil data dari API.
6. Buat struktur folder seperti di atas.
7. Kembangkan komponen individual untuk fungsi tertentu, seperti pencarian film atau tampilan detail film.
8. Tambahkan routing dengan React Router untuk navigasi antar halaman (beranda dan detail film).
9. Gunakan file CSS terpisah untuk setiap komponen agar gaya lebih terorganisir.
10. Jalankan aplikasi di localhost untuk memastikan semua fitur berfungsi dengan baik.

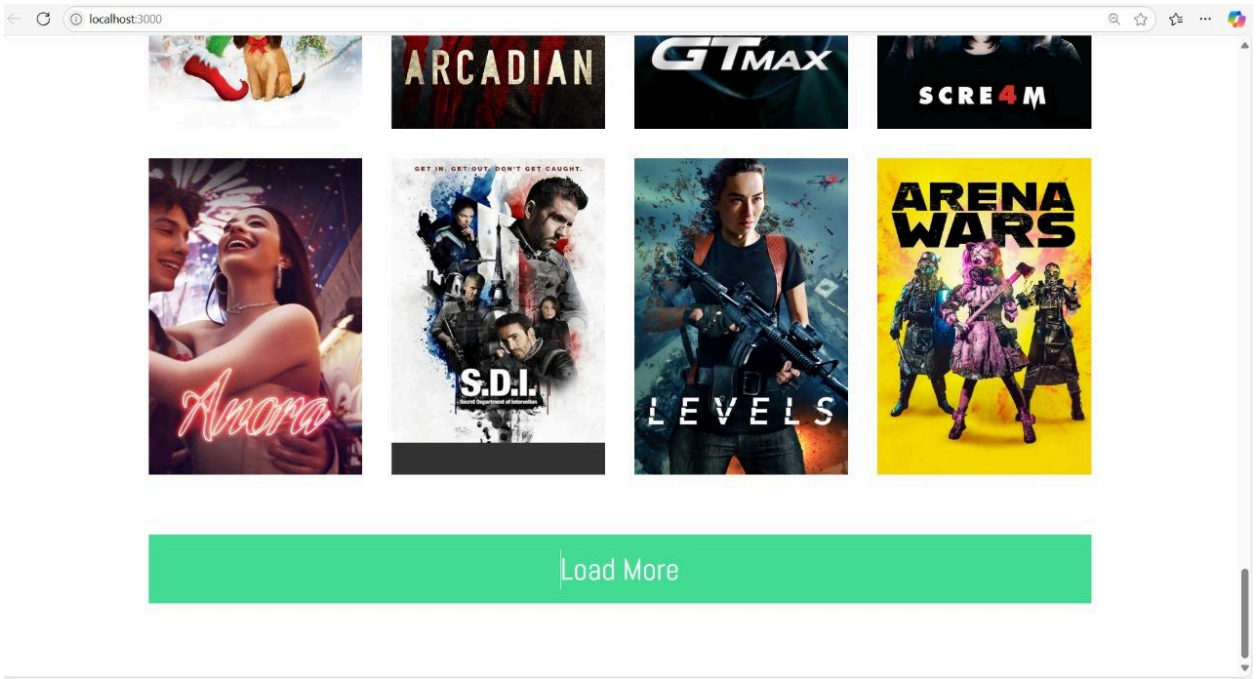
D. User Intrface



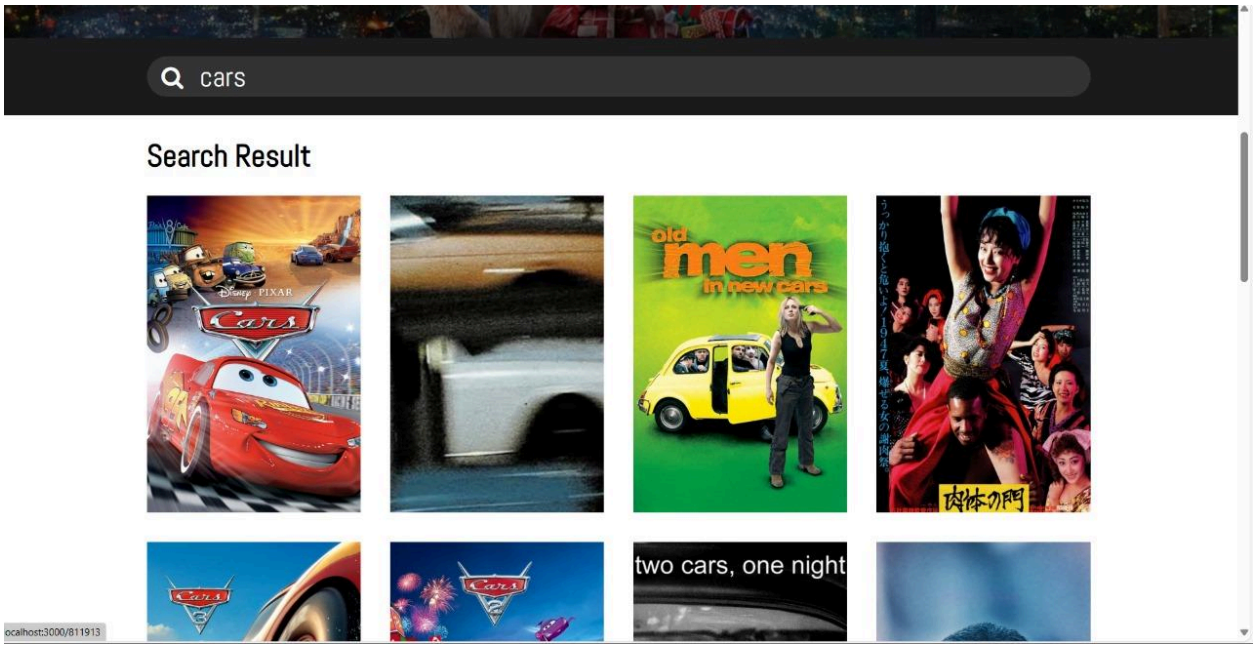
Gambar 1. Beranda



Gambar 2. Daftar film



Gambar 3. Fitur Load More



Gambar 4. Hasil Pencarian

E. Jobdesk Anggota

No	Nama Anggota	Rincian Job Desk
1.	Shafira Faiz Aulia W	1. Menambahkan detail Halaman film/serial seperti : - Poster film. - Sinopsis, genre, rating, dan informasi lainnya. - Daftar aktor dan kru. 2. Membuat animasi atau transisi sederhana.. 3. Menyusun Laporan Akhir dari Final Projek Pem.Terintegrasi
2.	Muhammad Rafli Alfarisqi	1. Menangani interaksi pengguna seperti pencarian film, filter, dan navigasi halaman. 2, Melakukan integrasi API 3. Mengambil data dari API. 4. Mengatur state global untuk menyimpan data film/serial.

		5. Membuat komponen logis seperti: - Search Bar: Untuk mencari film/serial. - Pagination/Load More Button: Untuk memuat lebih banyak data.
3.	Alfatur Rabbani	1. Membuat struktur dasar proyek (setup React, folder, dan file yang diperlukan). 2. Mengembangkan komponen utama untuk tampilan aplikasi seperti: 3. Halaman Beranda (Home Page): Menampilkan daftar film/serial populer. 4. Header dan Navigasi (Navigasi ke halaman Home, Detail Film, dll.). 5. Komponen layout seperti grid, carousel, atau kartu film.

F. Link Project

https://github.com/kingzturch/newuas