KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 1, Januari (2025), pp. 187-196

Implementasi *Fetch API* dalam pengembangan Backend Website Daftar Film dengan Next.JS

Irvan Dhimas Maulana[1], Yeremia Alfa Susetyo[2]
Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Indonesia
Email: 672021186@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

Abstract

This study aims to improve the performance of a film list website by implementing the Next.JS framework along with its Fetching API to solve challenges in search efficiency and providing accurate and up-to-date information for users. The amount of outdated and irrelevant data on film websites often prevents users in finding current movie information. This research develops an application using Next.JS, integrated with the Fetch API, to dynamically retrieve data using Server-Side Rendering (SSR) and Static Site Generation (SSG), improving communication between server and client, while providing a responsive and SEO-friendly user experience. Testing results with Lighthouse and Chrome DevTools show improved performance, with an application score of 92 on the Vercel platform and 82 on the Local side. Cache optimization on Vercel also reduced data transfer size from 2.2 MB to 0.27 MB, significantly speeding up load times and stabilizing the application. These results indicate that the application successfully delivers relevant and up-to-date information with high speed and stable performance. However, this study is limited in terms of testing devices and focuses only on the Vercel hosting platform.

Keywords: *Next.JS, Server-Side Rendering (SSR), Fetch API, SEO, Website Performance, Caching, Lighthouse, Chrome DevTools.*

Abstrak

Penelitian ini bertujuan untuk meningkatkan performa website daftar film dengan menerapkan framework Next.JS dan Fetching API milik Next.JS, yang mengatasi tantangan efektivitas dalam pencarian dan relevansi informasi yang akurat dan terkini bagi pengguna. Banyaknya data yang tidak relevan dan kurang update pada situs film menyulitkan pengguna dalam mencari informasi-informasi terkini seputar film yang mereka cari. Penelitian ini mengembangkan aplikasi dengan Next.JS dilengkapi dengan Fetch API untuk mengambil data yang dinamis dengan menggunakan Server-Side Rendering (SSR) dan Static Site Generation (SSG) untuk meningkatkan komunikasi antara server dengan klien, memberikan pengalaman responsif dan SEO-friendly bagi pengguna. Dari hasil pengujian menggunakan Lighthouse dan Chrome Devtools menunjukkan peningkatan performa dengan skor aplikasi di 92 pada platform Vercel dan 82 untuk sisi Lokal. Optimalisasi cache di Vercel juga mengurangi ukuran transfer data nya dari 2,2 MB menjadi 0,27 MB, yang di mana mempercepat waktu load dan membuat aplikasi menjadi lebih stabil. Hasil ini merepresentasikan bahwa aplikasi yang dikembangkan berhasil menampilkan informasi terkini dan relevan dengan kecepatan yang tinggi dan performa yang stabil. Namun, penelitian ini memiliki keterbatasan dalam cakupan perangkat pengujian serta fokus platform hosting yang terbatas di Vercel.

Kata Kunci: *Next.JS, Server-Side Rendering (SSR), Fetch API, SEO, Performa Website, Cache, Lighthouse, Chrome Devtools.*

ISSN: 2720-992X

1. Pendahuluan

Film adalah sebuah media penyaluran yang memiliki bentuk komunikasi audio serta visual serta memiliki kegunaan sebagai sebuah pembelajaran ataupun motivasi melalui pesan yang disampaikan di dalamnya [1]. Tidak hanya diartikan sebagai karya seni saja, film juga bisa diartikan sebagai praktik sosial. Dengan banyaknya film yang beredar saat ini dan juga genre yang begitu luas [2]. Film juga selalu berkembang dari zaman ke zaman dengan Teknik dan teknologi yang berbeda-beda dengan kreativitas yang berbeda-beda [3].

Perkembangan teknologi saat sangat signifikan hingga sampai saat ini yang membantu banyak aspek di kehidupan sehari-hari sehingga berdampak positif. Banyaknya situs website film yang tersebar luas di internet sering kali membuat kebingungan bagi pengguna yang sedang ingin mencari sumber yang dapat diandalkan karena minimnya informasi membuat efektivitas pencarian film menjadi menurun. Informasi ini merupakan sumber yang sangat penting bagi website daftar film. Oleh karena itu, dibutuhkan teknologi yang mumpuni sehingga data yang di dapat adalah data yang terkini dan dinamis. Dalam penelitian ini akan membuat pemanfaatan *Fetch API* dalam sebuah website untuk menyelesaikan permasalahan yang ada sesuai dengan kebutuhan sistem.

Dalam penanggulangan masalah yang terjadi pada website daftar film, *Fetch API* dari *framework javascript* yaitu Next.JS menjadi solusi efektif dikarenakan teknologi ini sangat efektif untuk menangani adanya informasi-informasi yang kurang relevan seperti data yang tidak lagi dibutuhkan atau tidak berguna untuk pengguna saat ini, misalnya detail lama atau tidak terkini dari sebuah daftar film yang telah mengalami perubahan dan kurang terkini, serta berguna untuk mengambil data yang terus berubah dan tidak menetap dari sebuah server. Dengan menggunakan *Fetch API*, komunikasi antara klien dan server dapat dilakukan secara dinamis tanpa perlu melakukan *refresh* halaman secara keseluruhan. Ini memungkinkan aplikasi untuk mengambil data yang terus berubah langsung dari server saat diperlukan, seperti informasi terbaru tentang film atau status terbaru dari sebuah proyek, tanpa membebani pengalaman pengguna dengan memaksa mereka untuk memuat ulang halaman secara manual [4]. Dengan demikian, implementasi *Fetch API* tidak hanya efisien dalam mengatasi masalah informasi yang kurang relevan, tetapi juga memungkinkan aplikasi untuk tetap terhubung dengan data yang paling terbaru. Dengan implementasi *Fetch API* tersebut, dapat memudahkan komunikasi antara klien dengan server dan masalah yang ada pada website daftar film dapat terselesaikan dengan efisien dan efektif. Terdapat *getServerSideProps* dan *getStaticProps* yang memungkinkan penggunaan *Fetch API* secara langsung di *Server-Side Rendering* (SSR) dan *Static Site Generation* (SSG) [5].

Penerapan *Fetch API* di dalam backend sebuah website daftar film menggunakan Next.JS muncul sebagai solusi yang efektif. Teknologi ini menghadirkan kemampuan yang sangat berguna dan efisien dalam menangani informasi-informasi yang kurang relevan dan kurang terkini serta berubah-ubah seiring berjalannya waktu. Next.JS telah mampu mengambil data yang terus berubah dari server tanpa *refresh* halaman. Implementasi *Fetch API* melalui *getServerSideProps* dan *getStaticProps* memfasilitasi komunikasi *client-server* yang efisien dan menyelesaikan masalah website daftar film secara efektif.

Dari penjelasan di atas, dengan menerapkan *Fetch API* dengan *framework* Next.JS ini, efisiensi pencarian film meningkat dengan data terkini dan relevan. Hal ini meningkatkan kualitas informasi yang disajikan dalam website secara *real-time*, *SEO-friendly*, dan juga fleksibilitas.

2. Metodologi Penelitian

Pada penelitian yang berjudul "Penerapan Rest API untuk Sistem Informasi Film Secara Daring" disebutkan bahwa penelitian ini bertujuan untuk merancang dan

mengimplementasikan sistem aplikasi berbasis website yang memanfaatkan struktur navigasi campuran dan TMDb API untuk menyediakan informasi film kepada pengguna. Sistem Informasi Film Secara Daring dibangun untuk menyederhanakan penyimpanan data film. Tujuannya adalah untuk mengembangkan sistem informasi yang memudahkan pengelolaan jenis, genre, dan tipe film secara efisien menggunakan bahasa pemrograman *JavaScript* dan CSS *Grid* serta *FlexBox* [1]. Namun *Javascript* native memiliki beberapa kelemahan termasuk keterbatasan dalam mengelola asynchronous, yang dapat menyulitkan dalam mengelola alur eksekusi kode. Sementara itu, kelemahan CSS *Grid* dan *FlexBox* adalah kurangnya dukungan pada beberapa browser lama, sehingga dapat menyebabkan tampilan yang tidak konsisten pada berbagai platform. Oleh karena itu, di penelitian ini Next.JS bisa untuk membangun aplikasi web yang lebih interaktif menggunakan *framework* Tailwind, *real-time*, dan modern dengan dukungan *Server-Side Rendering* yang kuat. Salah satu kelemahan jurnal ini adalah kurangnya pembahasan secara khusus mengenai penggunaan *Fetch API* dengan Next.JS. Jurnal ini lebih fokus pada implementasi *Fetch API* dengan bahasa pemrograman *JavaScript* dan CSS *Grid* serta *FlexBox* untuk pembentukan layout aplikasi penyedia informasi film berbasis website. Sementara itu, Next.JS menawarkan kemudahan dalam melakukan *fetching* data dengan fungsi *built-in* untuk *Server-Side Rendering*, serta dukungan *prefetching* data secara otomatis untuk meningkatkan performa aplikasi.

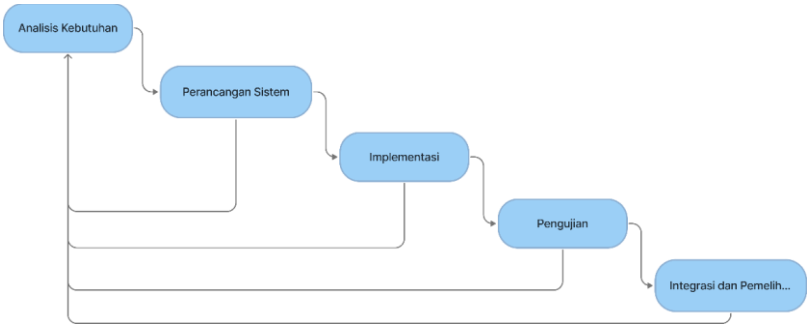
Pada penelitian yang berjudul "Implementasi Progressive Web Apps Pada Website GetHelp Menggunakan Next JS" menyebutkan bahwa implementasi Progressive Web Apps (PWA) pada platform GetHelp menggunakan *framework* Next.JS bertujuan memungkinkan akses melalui layar utama perangkat seluler seperti aplikasi Android dan menyediakan fungsionalitas secara luring. Penelitian ini bertujuan mengimplementasikan teknologi *Progressive Web Apps* (PWA) pada website *GetHelp* dengan menggunakan *framework* Next.JS untuk memungkinkan akses website melalui beranda di *smartphone* seperti aplikasi Android dan *offline*. Implementasi melibatkan instalasi PWA Next.JS, pembuatan web app *manifest*, registrasi service worker, dan evaluasi PWA. Evaluasi menunjukkan bahwa layanan website *GetHelp* dapat diakses dengan baik pada berbagai perangkat *mobile* Android. Selain itu, evaluasi performa sistem menggunakan *Lighthouse* menunjukkan peningkatan performa sebesar 23%, sedangkan dengan *GTMetrix* performa website meningkat 14% setelah implementasi PWA dengan Next.JS [6]. Dalam penelitian tersebut, tidak membahas secara mendalam atau tidak menyertakan informasi tentang pembahasan *Fetching API* dalam implementasi *Progressive Web Apps* (PWA) pada website *GetHelp* dengan *framework* Next.JS. Hal ini dapat menjadi kekurangan dalam konteks penelitian yang berfokus pada pengembangan website dengan Next.JS yang melibatkan *Fetch API* untuk memastikan kelengkapan dan efisiensi dalam penggunaan teknologi tersebut.

Pada penelitian yang berjudul "Front End Learning Management System Development Using The Next JS *Framework*" menyebutkan bahwa pembangunan sistem komputerisasi dalam metode pembelajaran, dengan salah satu sistem yang diterapkan adalah sistem pendukung keputusan yang dapat membantu perusahaan dalam pengambilan keputusan menggunakan data dan model yang ada. Media pembelajaran saat ini memainkan peran penting dalam pendidikan di era digital, di mana teknologi dan informasi berperan besar. *Learning Management System* (LMS) digunakan sebagai aplikasi pembelajaran *online*, namun sering kali memiliki keterbatasan fitur dan interaktivitas. Pembangunan LMS bertujuan untuk menciptakan platform pembelajaran yang lebih interaktif dan komprehensif. Analisis sistem dilakukan untuk mengidentifikasi kelemahan dan kelebihan melalui pendekatan metode *waterfall*. Hasilnya adalah LMS dengan fitur lengkap dan inovatif, seperti konferensi video, dokumen & berkas, obrolan grup, dan pemeriksaan otomatis [7]. Dalam penelitian tersebut, Next.JS memiliki kemampuan dalam memproses data dengan lebih cepat selama proses *rendering* merupakan faktor kunci dalam

meningkatkan performa dan kecepatan antarmuka aplikasi web. Melalui Next.JS, sistem dapat menghadirkan informasi kepada pengguna dengan lebih cepat dan efisien.

Penelitian ini mengadopsi metodologi *Waterfall* dalam pengembangan aplikasi website daftar film dengan menggunakan Next.JS. Metodologi *Waterfall* dipilih karena pendekatan ini memberikan kerangka kerja yang terstruktur dan teratur, memfasilitasi pengembangan bertahap yang jelas dan terukur. Pada Gambar 1 adalah tahapan-tahapan yang dilalui dalam penelitian ini. Metode *Waterfall* menyusun tahapan pengembangan secara linier, dimulai dari analisis kebutuhan, perancangan sistem, implementasi, pengujian, hingga pemeliharaan. Pendekatan ini memberikan kerangka kerja yang terstruktur untuk memastikan setiap fase pengembangan dilakukan secara sistematis dan mengikuti urutan yang jelas, sesuai dengan kebutuhan proyek pengembangan aplikasi website daftar film menggunakan Next.JS [8].

Berdasarkan Gambar 1 di atas, Pada tahap ini adalah tahap pengumpulan kebutuhan yang mengenai fitur-fitur yang dibutuhkan oleh pengguna dan admin. Dalam website yang dikembangkan, fitur utama yang disediakan untuk pengguna meliputi akses ke daftar film atau anime, kemampuan *login* menggunakan akun *Google* dan *GitHub*, kemampuan untuk menyimpan film atau anime ke dalam koleksi pribadi pengguna, kemampuan untuk memberikan komentar di film atau anime tertentu, dan juga mencari film atau anime yang pengguna sukai. Sementara itu, fitur untuk admin mencakup manajemen akun pengguna, pengelolaan koleksi film atau anime pengguna, pengelolaan komentar pengguna, serta manajemen film atau anime seperti penambahan, pengubahan, dan penghapusan daftar film atau anime yang akan ditampilkan di halaman pengguna.



Gambar 1. Alur Penelitian menggunakan Metode *Waterfall*

Lalu di perancangan sistem, dilakukan desain arsitektur aplikasi. Desain arsitektur mencakup penerapan *Server-Side rendering* (SSR) untuk meningkatkan performa dan pengalaman pengguna. Dengan SSR, halaman-halaman situs akan di render di server sebelum dikirim ke *client*, sehingga dapat mempercepat waktu muat dan meningkatkan *SEO*. Alur kerja *Fetch API* dalam Next.JS digunakan untuk melakukan *fetching* data secara dinamis dari API eksternal dan dari server. *Database MongoDB* digunakan untuk menyimpan data komentar dan koleksi pengguna, sementara *ORM Prisma* digunakan untuk menghubungkan Next.JS dengan *MongoDB*, memastikan integrasi yang efisien dan aman antara aplikasi dan *database* [9].

Tahap implementasi mencakup yang pertama yaitu pengembangan *Backend* menggunakan Next.JS dan penerapan SSR melalui fungsi *getServerSideProps*. Lalu setelah itu penggunaan *Prisma ORM* sebagai jembatan dengan *database MongoDB*. Setelah itu dilakukan pengambilan data secara dinamis dari server dilakukan melalui *Fetch API* untuk memastikan data yang diambil selalu terkini dan relevan. Selain itu, terdapat pengembangan fitur *login* menggunakan *Google* dan *GitHub OAuth2* untuk autentikasi pengguna. Selain itu, terdapat implementasi fitur penyimpanan koleksi dan komentar, termasuk integrasi fitur tersebut dengan *database MongoDB* melalui *ORM Prisma*.

Tahapan ini melibatkan serangkaian pengujian yang berorientasi pada validitas logika dan fungsi perangkat lunak, serta memeriksa kesesuaian setiap komponen untuk memastikan bahwa keluaran yang dihasilkan sesuai dengan yang diharapkan. Pengujian dilakukan secara menyeluruh mencakup beberapa aspek penting. Pertama, pengujian fungsional dilakukan untuk memverifikasi bahwa semua fitur aplikasi berjalan sesuai dengan yang diharapkan, memastikan setiap komponen dan fungsi beroperasi dengan benar. Kedua, pengujian integrasi dilakukan untuk memastikan bahwa *Server-Side rendering* (SSR), *ORM Prisma*, *MongoDB*, dan *API* yang sudah di *fetch* bekerja bersama-sama tanpa masalah, menjamin bahwa semua bagian sistem dapat berkomunikasi dan beroperasi secara sinergis. Ketiga, pengujian performa dilakukan untuk memastikan aplikasi dapat menangani beban kerja yang diinginkan dan memiliki waktu respons yang cepat, menjaga kualitas pengalaman pengguna di bawah berbagai kondisi.

Setelah pengujian selesai dan aplikasi dinyatakan siap digunakan, dilakukan integrasi dengan sistem yang ada. Tahap pemeliharaan melibatkan pemantauan aplikasi secara terus-menerus untuk mengidentifikasi dan memperbaiki masalah yang muncul, serta melakukan pembaruan dan peningkatan fitur sesuai kebutuhan. Integrasi ini memastikan bahwa aplikasi tetap berfungsi dengan baik dan dapat ditingkatkan seiring waktu untuk memenuhi kebutuhan pengguna yang berkembang.

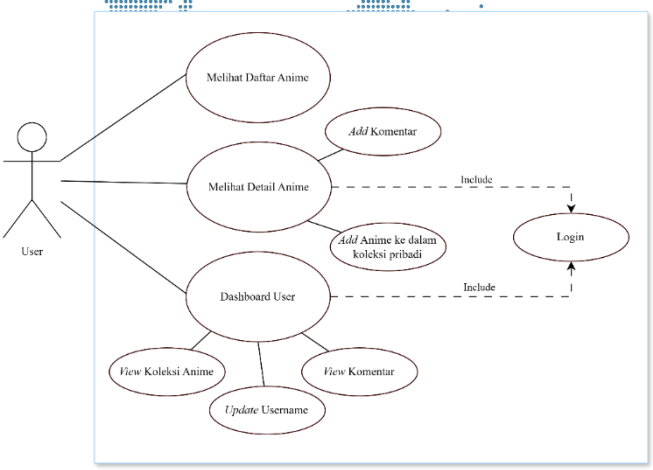
3. Hasil dan Pembahasan

Pada penelitian ini, implementasi menggunakan Next.JS dengan memanfaatkan *Fetching API* dan *Server-Side Rendering* (SSR) digunakan untuk membuat dua aplikasi web berbeda, yaitu ADMIN dan USER. Dalam aplikasi ini, beberapa komponen penting yang digunakan antara lain yaitu *Header*.js untuk menampilkan *header* pada semua halaman di sisi USER dan ada *Collection*List.js untuk menampilkan daftar koleksi anime pengguna di sisi ADMIN.

Gambar 2. Use Case Diagram Admin

Pada Gambar 2 menjelaskan bahwa administrator memiliki akses *login* ke *Dashboard* Admin menggunakan *email* dan *password* yang telah terdaftar, dengan autentikasi melalui *NextAuth* di Next.JS. Di dalam *dashboard*, admin dapat mengelola daftar akun pengguna, termasuk mengedit informasi seperti *username*, *email*, dan *password*, atau menghapus akun pengguna jika diperlukan. Selain itu, admin dapat mengakses dan mengelola koleksi anime yang dibuat oleh pengguna. Mereka juga bisa melihat dan menghapus komentar

yang ditambahkan oleh pengguna di halaman detail anime. Admin memiliki kontrol untuk memilih dan mengatur anime yang ditampilkan ke pengguna, memberi mereka kewenangan untuk menambah atau menghapus anime dari tampilan pengguna.



Gambar 3. *Use Case Diagram* User

Pada Gambar 3 menjelaskan bahwa pengguna dapat *login* menggunakan akun *Google* atau *GitHub* melalui *NextAuth*, yang memanfaatkan autentikasi *OAuth2* dari kedua penyedia tersebut. Di halaman beranda, pengguna bisa melihat daftar anime populer, anime musim Spring 2024, dan anime yang paling banyak ditonton hari ini. Selain itu, pengguna dapat mencari anime tertentu melalui bilah pencarian di bagian atas halaman. Saat mengeklik salah satu anime, pengguna akan diarahkan ke halaman detail yang menyajikan informasi seperti sinopsis, genre, skor, dan trailer, serta opsi untuk menambahkan komentar dan menyimpan anime ke koleksi pribadi (jika pengguna sudah *login*). Di halaman detail, pengguna dapat menulis komentar yang tersimpan bersama komentar pengguna lain, dengan hak untuk menghapus komentar mereka sendiri namun tidak komentar dari pengguna lain. Koleksi anime yang ditambahkan oleh pengguna akan disimpan di *database* dan ditampilkan di halaman *Collection*, memungkinkan pengguna untuk mengakses ke anime yang sudah mereka tambahkan. Pengguna juga bisa mengubah *username* mereka di halaman profil, dan pembaruan ini langsung tercatat di *database*.

Server-Side Rendering (SSR) adalah sebuah fitur utama yang diaplikasikan di kedua aplikasi di atas, baik ADMIN maupun USER, untuk meningkatkan performa dan *Search Engine Optimization* (*SEO*). Cara kerja SSR yaitu dengan cara me-render halaman di sisi pertama kali lalu dikirim ke browser yang sudah berupa HTML lengkap, hal ini sangat memengaruhi strategi pemuatan halaman dan dampaknya terhadap performa aplikasi serta peringkat *SEO*. Salah satu keuntungan utama *Server-Side Rendering* (SSR) di Next.JS adalah optimalisasi *SEO* yang lebih baik dan efisien. Secara default, Next.JS memiliki fitur dukungan untuk *Server-Side Rendering* (SSR). Dalam kasus SSR, halaman yang dihasilkan dalam HTML dapat dirayapi oleh bot *SEO*, sehingga kemungkinan besar halaman tersebut akan muncul dalam pencarian. Selain itu, Next.JS telah membuatnya sangat mudah untuk mengembangkan judul dan kata kunci untuk setiap halaman yang sedang dikembangkan. Secara umum, Next.JS berkontribusi banyak dalam hal kinerja dan aksesibilitas sehubungan dengan bot pencarian, kedua faktor ini penting dalam *SEO* [10].

Real-time itu sendiri adalah kemampuan sistem untuk memantau dan mengolah data secara langsung saat data tersebut dihasilkan, sehingga memungkinkan respons yang cepat terhadap perubahan yang terjadi [11]. Di aplikasi ADMIN, fitur SSR ini membuat pengambilan data dapat dilakukan secara *real-time*, seperti data pengguna, koleksi, dan komentar dari *database*. Setiap kali administrator membuka halaman, data langsung

diambil dari server dan ditampilkan tanpa perlu melakukan operasi manual seperti *refresh* halaman. Ini memastikan bahwa data yang disajikan selalu yang ter-update, karena server selalu me-render halaman terbaru yang diambil dari *database*. Begitu juga pada aplikasi USER, di mana SSR digunakan untuk menampilkan daftar anime populer secara dinamis.

Terdapat *getServerSideProps* di dalam Next.JS yaitu untuk membangun permintaan di server sebelum halaman di-render. Setiap komponen file dalam Next.JS bersifat server komponen, sehingga konten seperti HTML dan *JavaScript* akan dibangun dan diproses di server [12]. Proses ini juga melibatkan tahap hydration, yang memungkinkan caching berjalan baik di sisi server maupun klien. Fitur ini digunakan untuk mengambil data di sisi server setiap kali halaman diakses. Fungsi ini dijalankan di sisi server pada setiap *request* dan memungkinkan juga untuk mendapatkan data yang ingin ditampilkan di halaman sebelum di-render.

Kode program 1 *GetServerSidePops pada Dashboard Admin*

```
import CommentList from "@/components/Dashboard/CommentList";
import prisma from "@/services/prisma";

const Page = async () => {
  const comment = await prisma.comment.findMany(); // Fetch all users from the database

  return (
    <section className="w-full px-4 mt-4">
      <h1 className="text-2xl font-bold text-color-secondary">Admin Dashboard - Manage
Comments</h1>
      <CommentList comment={comment} />
```

Context API memungkinkan data dibagikan di antara komponen tanpa harus meneruskannya melalui setiap tingkat komponen. Dengan ini, komponen yang membutuhkan data bisa langsung mengaksesnya, membuat alur data lebih sederhana dan efisien [13]. *Context API* adalah fitur yang digunakan untuk mengelola konfigurasi global, sehingga komponen-komponen yang ingin mengaksesnya menjadi lebih efisien dalam pengambilan data serta waktu dan mudah karena sudah menjadi variabel global.

Middleware adalah perangkat lunak yang berfungsi sebagai perantara yang menghubungkan berbagai aplikasi atau layanan yang tidak langsung terhubung satu sama lain, sering kali *middleware* menggunakan arsitektur *Service-Oriented Architecture* (SOA) untuk menyediakan layanan analisis dan pertukaran data dalam infrastruktur IT yang kompleks. [14]. *Middleware* berfungsi untuk melakukan intervensi pada *request* sebelum mencapai *endpoint* yang diinginkan. Jadi ketika ingin mengakses *endpoint* tertentu maka akan dihalang terlebih dahulu jika sudah masuk ke *middleware*.

Dalam skema pengujian ini terdapat beberapa *tools* dan metode pengujian. *Tools* yang digunakan yaitu *Lighthouse* dan *Chrome DevTools* dengan mengukur *SEO*, Performance, Accessibility, dan Best Practices nya untuk di *Lighthouse*. Sementara dari sisi *Chrome DevTools* digunakan untuk menguji bagian *Network* dan Performance. Di sini terdapat 2 Environment yaitu Lokal dan di *Vercel*. *Lighthouse* merupakan alat audit website otomatis yang bersifat *open-source*, alat ini digunakan untuk menilai dan meningkatkan kualitas suatu situs web. Alat ini melakukan evaluasi terhadap performa, aksesibilitas, dan lain-lain [15].

Tabel 1. Hasil Pengujian *Lighthouse*

Matriks	**Lokal**	***Vercel***
Performance	82	92
Accessibility	92	92
Best Practices	100	96
SEO	91	91

Pada segi *Performance* dari *Lighthouse*, untuk Lokal memberikan skor 82 dan di *Vercel* 92. Dari hasil tersebut, skor untuk metrik inti seperti First Contentful Paint (FCP), Speed Index, Largest Contentful Paint (LCP), Time to Interactive, dan Cumulative Layout Shift (CLS) merefleksikan performa yang lebih tinggi di platform *Vercel* karena optimilalisasi dari CDN, serta infrastruktur cache yang lebih baik. Dan untuk sisi Lokal dipengaruhi oleh perangkat yang digunakan, keterbatasan sistem, dan jaringan yang digunakan sehingga berpengaruh terhadap performa yang dihasilkan menjadikannya sedikit lebih rendah.

Pada segi Aksesibilitas, sama-sama memiliki nilai yang tinggi yaitu di skor 92 di Lokal dan juga *Vercel*. Ini menunjukkan bahwa aplikasi ini sudah cukup ramah bagi pengguna yang memiliki kebutuhan aksesibilitas khusus. Seperti contoh: Penggunaan teks alternatif pada Gambar, struktur *heading* yang sesuai, serta label yang jelas untuk elemen form nya.

Pada segi *Best Practices*, pada Lokal menghasilkan skor 100 sementara di *Vercel* menghasilkan skor 96. Ini merefleksikan bahwa aplikasi ini sudah memenuhi aspek utama seperti penggunaan HTTPS, pengaturan cookie aman, struktur HTML yang baik, menghindari API yang sudah usang. Skor untuk *Vercel* memang lebih rendah karena berasal dari konfigurasi tambahan yang otomatis terdeklarasi dari platfrom *Vercel* itu sendiri yang mempengaruhi validitas HTML nya.

Pada segi *SEO*, menghasilkan skor 91 di kedua sisi yaitu di Lokal dan *Vercel* yang menunjukkan bahwa aplikasi ini sudah cukup optimal dengan konfigurasi mesin pencari. Seperti meta tags yang relevan, struktur URL yang baik, dan *heading* yang hierarkis.

Tabel 2. Total Load per *Request* pada LOKAL (ms)

Matriks	*Queueing*	*Stalled*	*Wait*	*Download*
Request 1	20,59	27,3	0,53	5,25
Request 2	38,58	14,22	0,97	3,93
Request 3	32,67	13,74	0,91	4,46
Request 4	38,75	14,03	0,93	4,17
Request 5	33	13,25	0,9	4,97
Request 6	67,06	15,93	0,37	15,13
Request 7	72,25	22,13	0,89	21,76
Request 8	72,69	21,72	0,91	20,66
Request 9	73,25	21,16	0,89	17,28
Request 10	74,72	19,67	0,9	17,66

Tabel 3. Total Load per *Request* pada *VERCEL* (ms)

Matriks	*Queueing*	*Stalled*	*Wait*	*Download*
Request 1	20,59	27,3	0,53	5,25
Request 2	38,58	14,22	0,97	3,93
Request 3	32,67	13,74	0,91	4,46
Request 4	38,75	14,03	0,93	4,17
Request 5	33	13,25	0,9	4,97
Request 6	67,06	15,93	0,37	15,13
Request 7	72,25	22,13	0,89	21,76
Request 8	72,69	21,72	0,91	20,66
Request 9	73,25	21,16	0,89	17,28
Request 10	74,72	19,67	0,9	17,66

Lalu pada pengujian pada *Chrome Devtools*, pengujian di berbagai kondisi *network* menunjukkan perbedaan signifikan antara performa di platform lokal dan *Vercel*, terutama dalam pengelolaan cache dan jumlah *request*. Pada pengaturan *Network* (Enable Cache,

No Throttling), *Vercel* menangani 116 *request* dibandingkan 42 *request* di lokal, menunjukkan adanya tambahan cache atau permintaan ulang yang lebih banyak di *Vercel*. Meskipun begitu, efisiensi caching di *Vercel* menonjol, terlihat dari ukuran transfer data yang hanya sebesar 0,27 MB dibandingkan 2,2 MB pada server lokal, menandakan optimalisasi transfer yang lebih baik. Dengan cache aktif, waktu download pada server lokal mengalami penurunan signifikan, tetapi *Vercel* tetap mempertahankan waktu unduhan rendah dan lebih stabil secara keseluruhan. Hasil ini menunjukkan bahwa optimasi *Vercel* melalui cache tidak hanya mengurangi beban transfer data, tetapi juga mempertahankan konsistensi waktu download, bahkan ketika jumlah permintaan lebih tinggi.

4. Kesimpulan

Dari penelitian dan implementasi yang telah dilakukan, dapat ditarik kesimpulan bahwa *framework* Next.JS memberikan peningkatan signifikan pada performa aplikasi terutama pada kecepatan pemuatan konten dengan fitur *Server-Side Rendering* (SSR) milik Next.JS dan efisiensi pengelolaan jaringan pada platform *Vercel*. Dari hasil di atas, aplikasi berhasil menampilkan konten lebih cepat dan responsif hal ini dapat dilihat dari skor *Lighthouse* pada segmen performa mencapai 92 di *Vercel* dan 82 pada Lokal dikarenakan kinerja dari platform *Vercel* yang lebih optimal dalam kompresi data dan caching-nya bahkan dalam jaringan lambat seperti di jaringan 3G. Saat cache diaktifkan dan tanpa throtlling, misalnya, *Vercel* hanya membutuhkan 0,27 MB jika dibandingkan dengan sisi Lokal yang memerlukan 2,2 MB. Disini menunjukkan bahwa optimalisasi jaringan dan caching pada *Vercel* dengan efektif menurunkan kebutuhan transfer data, mengurangi waktu download, serta membuat performa aplikasi menjadi stabil.

Pada hasil pengujian performa menunjukkan bahwa *Vercel*, memiliki keunggulan dalam pengelolaan *network* dan optimalisasi data melalui sistem kompresi dan caching yang efisien. Dalam kondisi jaringan lambat, seperti 3G, *Vercel* mampu menangani lebih banyak *request* dengan ukuran transfer data lebih rendah, sehingga mempercepat waktu loading dan mengurangi beban transfer data di perangkat pengguna. Pengaktifan caching di sisi *Vercel* juga memberikan dampak positif dalam menurunkan waktu download pada beberapa *request* dan juga menjaga kestabilan performa aplikasi meskipun dalam lingkungan dengan bandwidth terbatas. Selain itu, saat diuji dengan pembatasan CPU yang diatur menjadi 6x slow-down, *Vercel* menunjukkan keunggulan dalam scripting dan *rendering* yaitu dengan mengelola proses yang lebih berat dengan efisiensi yang lebih tinggi ketimbang di sisi Lokal, sehingga performa aplikasi tetap cepat di berbagai perangkat, termasuk pada spesifikasi rendah atau di bawah beban CPU tinggi.

Namun, terdapat keterbatasan dalam pengujian ini. Pengujian ini belum memperhitungkan variasi perangkat yang lebih luas dalam kasus *real-time*, yang di mana akan mempengaruhi performa aplikasi. Selain itu, pengujian ini fokus pada platform *Vercel* dengan sistem dan optimalisasi dari *Vercel* itu sendiri sehingga hasilnya belum mencakup perbandingan dengan platform *hosting* lainnya. Untuk penelitian selanjutnya, diharapkan untuk menguji performa Next.JS pada platform *hosting* yang berbeda serta menguji aplikasi dengan fitur yang lebih kompleks di berbagai perangkat agar kinerja dan skalabilitas aplikasi dalam kondisi *real-time* lebih tervisualisasikan

Daftar Pustaka

[1] S. N. Yanti Dan E. Rihyanti, "Penerapan Rest Api Untuk Sistem Informasi Film Secara Daring," *Jurnal Informatika Universitas Pamulang*, Vol. 6, No. 1, Mar 2021, Doi: 10.32493/Informatika.V6i1.10033.

[2] A. Muni Dan K. Ihwan, "Perangcangan Sistem Informasi Film Berbasis Web," *Jurnal Teknik Industri Unisi) Penulis: Judul Artikel*, Vol. 5, No. 2, Des 2021.

[3] H. Tusakdiah, "Analisis Nilai Pendidikan Karakter Dalam Film Sang Prawira Karya Onet Adithia Rizlan," Universitas Maritim Raja Ali Haji, Tanjung Pinang, 2022.

[4] T. A. Ningsih Dan I. M. Widiartha, "Pemanfaatan Api Dalam Menampilkan Data Dinamis Untuk Sistem Informasi Film," *Jurnal Elektronik Ilmu Komputer Udayana*, Vol. 12, No. 1, Agu 2023.

[5] R. Ardiyanto, E. Ardhianto, Dan C. Author, "Analisa Peformasi Metode *Rendering* Website: *Client* Side, Server Side, Dan Incremental Static Regeneration," *Computer Science (Co-Science*, Vol. 4, No. 1, Jan 2024.

[6] R. Jubhari Phie Joarno, M. Fajar, Dan A. Yunus, "Implementasi Progressive Web Apps Pada Website Gethelp Menggunakan Next.Js," *Jurnal Kharisma Tech*, Vol. 17, 2022, [Daring]. Tersedia Pada: Https://Jurnal.Kharisma.Ac.Id/Kharismatech/

[7] A. Baehaqi, M. S. Basit, R. E. Indrajit, Dan R. D. Kurniawan, "Front End Learning Management System Development Using The Nextjs *Framework*," *Jurnal Teknik Informatika (Jutif)*, Vol. 4, No. 4, Hlm. 899–911, Agu 2023, Doi: 10.52436/1.Jutif.2023.4.4.1273.

[8] M. Badrul Dan Kurniawati, "Penerapan Metode *Waterfall* Untuk Perancangan Sistem Informasi Inventory Pada Toko Keramik Bintang Terang," *Jurnal Prosisko*, Vol. 8, No. 2, Sep 2021.

[9] P. Paunikar, D. Londhe, S. Pawar, R. Phulari, S. Dondge, Dan R. Chauhan, "Building A *Real-Time* Web-Based Chat Application With Next.Js, *Mongodb*, And *Prisma* Orm: Enhancing User Experience Through Optimized Decision-Making," *Journal Of Basic Science And Engineering*, Vol. 21, No. 1, 2024.

[10] H. A. Jartarghar, G. Rao Salanke, A. A. Kumar, Dan S. Dalali, "React Apps With *Server-Side Rendering*: Next.Js," *Journal Of Telecommunication, Electronic And Computer Engineering*, Vol. 14, No. 4, Hlm. 25–29, 2022.

[11] R. Maududy Dan D. R. Nursamsi, "Pengembangan *Real-Time* Monitoring Dan Data Logging Berbasis Web Pada Proses Robot Painting Untuk Meningkatkan Efisiensi Produksi," *Informatics And Digital Expert (Index)*, Vol. 5, No. 2, Hlm. 89–94, 2023, [Daring]. Tersedia Pada: Https://E-Journal.Unper.Ac.Id/Index.Php/Informatics

[12] R. Sinulingga Dan I. M. Suartana, "Implementasi Server Side *Rendering* Pada Sistem Absensi Mahasiswa Berbasis Website," *Journal Of Informatics And Computer Science*, Vol. 5, No. 4, 2024.

[13] T. Le, "Comparison Of State Management Solutions Between Context Api And Redux Hook In Reactjs," Metropolia University Of Applied Sciences, 2021.

[14] Suprihadi, S. Wijono, Dan K. D. Hartomo, "Analysis Service Architecture Utilizing Middleware For Information Services Management Systems," *International Journal Of Applied Science And Engineering*, Vol. 21, No. 2, 2023, Doi: 10.6703/Ijase.202406_21(2).001.

[15] A. Adli, T. Wihayanti, Dan D. Witarsyah, "Top 5 E-Commerce Performance Analysis Using *Google Lighthouse* Matrix," *Proceedings Of The International Conference On Enterprise And Industrial Systems (Icoeins 2023)*, Hlm. 239–248, 2023, Doi: 10.2991/978-94-6463-340-5_21.