KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 1, Januari (2025), pp. 120-128

Pengembangan *Dashboard Admin* menggunakan *React JS* dan *Ant Design* pada Toko Berkat

Rafael Deandra Pradipta[1], Yeremia Alfa Susetyo[2]*
[1,2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Jawa Tengah, Indonesia
E-mail: 672021177@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

Abstract

The development of a web-based administrative dashboard for CV Berkat was conducted using the React JS framework and the Ant Design library. The application development process followed the waterfall method, ensuring that each phase, from requirements analysis to system maintenance, was fulfilled and completed systematically. This dashboard is designed to monitor sales data, product management, and employee reports with an attractive and interactive user interface. Based on testing result using Lighthouse, it was shown that the application performs more optimally when hosted compared to running locally, with assesment metrics such as Largest Contentful Paint (LCP) and Total Blocking Time (TBT) showing better results. The combination of React JS and Ant Design has proven to accelerate the development process thanks to the readily available components and support for the Single Page Application (SPA) method. Overall, this dashboard meets CV Berkat's needs for real-time and efficient data monitoring while also facilitating future feature development.

Keywords: *React JS, Ant Design, Administrative Dashboard, SPA, Performance Testing, Waterfall.*

Abstrak

Pengembangan dasbor administratif berbasis website untuk CV Berkat ini dilakukan menggunakan framework React JS dan library Ant Design. Proses pengembangan aplikasi dilakukan dengan mengikuti metode waterfall, sehingga setiap prosesnya mulai dari analisis kebutuhan hingga pemeliharaan sistem dapat terpenuhi dan tercapai secara sistematis. Dasbor ini dirancang untuk melakukan pemantauan terhadap data penjualan, manajemen produk, serta laporan karyawan dengan tampilan antarmuka yang menarik dan interaktif. Berdasarkan hasil pengujian menggunakan lighthouse, menunjukkan bahwa performa lebih optimal apabila aplikasi berjalan di hosting dibandingkan dengan lokal, matriks penilaian seperti Largest Contentful Paint (LCP) dan Total Blocking Time (TBT) lebih baik. Kombinasi React JS dan Ant Design terbukti mempercepat proses pengembangan berkat komponen yang sudah disediakan dan mendukung metode Single Page Application (SPA). Secara keseluruhan, dasbor ini sudah memenuhi kebutuhan CV Berkat dalam memantau data secara langsung dan efisien, serta memfasilitasi untuk adanya pengembangan fitur di masa mendatang.

Kata Kunci: *React JS, Ant Design, Dasbor Administratif, SPA, Pengujian Performa, Waterfall.*

1. Pendahuluan

Salah satu bentuk dari kemajuan teknologi informasi adalah adanya sebuah dasbor pemantauan data. Dasbor atau *dashboard* adalah sebuah tampilan antarmuka yang menyajikan informasi penting yang diperlukan oleh *user* untuk mencapai atau mengamati suatu tujuan tertentu. Informasi tersebut disusun dan disajikan dalam satu layar untuk memudahkan *user* dalam melihat. Sebuah *dashboard* berisi tampilan lengkap pada satu monitor atau satu halaman komputer yang berisi informasi krusial, memungkinkan untuk mengetahui hal-hal yang penting secara cepat [1].

ISSN: 2720-992X

Menurut (Malik S, 2005), *Dashboard* adalah aplikasi yang menyajikan berbagai informasi seperti matriks, tolok ukur, target, hasil, serta peringatan secara visual dengan efektif. *Dashboard* merupakan inovasi baru dalam manajemen informasi yang terus berkembang. *Dashboard* telah menjadi sarana eksekusi untuk sejumlah inisiatif penting yang sudah diterapkan oleh seluruh organisasi di dunia. Fungsi dari *dashboard* adalah untuk memberi peringatan kepada *user* ketika suatu matriks telah berada di luar batas yang diterima (*overload*). Dalam konteks ini peringatan terdiri dari aturan serta tindakan yang menambah poin penting pada penerapan *dashboard* di perusahaan dengan indikator visual yang membantu [2].

Tantangan utama yang sering dihadapi oleh tim developer perusahaan adalah kurangnya *dashboard* administratif yang lengkap untuk mengkoordinasikan dan menyederhanakan proses kerja. Di era digital yang terus berkembang, pengelolaan data dan informasi menjadi aspek yang krusial untuk mendukung proses bisnis. Salah satu bentuk implementasinya adalah *dashboard administrative* yang berfungsi sebagai alat bantu untuk *monitoring* dan menganalisis data secara *real-time* [3].

Ketiadaan *dashboard* yang lengkap dan menyeluruh tidak hanya mempengaruhi produktivitas, melainkan juga memperlambat pengambilan keputusan yang strategis. Tantangan ini semakin diperburuk oleh kebutuhan dari perusahaan untuk terus *update* dan beradaptasi dalam lingkungan bisnis yang bergerak secara dinamis dan penuh dengan persaingan bisnis. Saat ini, kemampuan untuk bertahan dan lebih unggul di tengah persaingan bisnis ini sulit dicapai. Pengetahuan mengenai teknologi informasi menjadi sangat penting bagi pengusaha, mengingat cepatnya perubahan dalam dunia bisnis dan teknologi informasi. Agar tetap dapat berkompetisi, organisasi atau perusahaan harus secara konsisten memantau dan mengukur kinerja mereka untuk memastikan bahwa tujuan yang mereka tetapkan tercapai. Proses pemantauan ini memerlukan data dan informasi mengenai keseluruhan aspek organisasi, sehingga membutuhkan sistem yang dapat diandalkan dan terpadu [4].

Oleh karena itu, *dashboard administrative* dibangun menggunakan *framework React Js* untuk memastikan *modularitas* dan pengorganisasian kode yang lebih baik. Penggunaan komponen di dalam *React Js* ini membantu menyederhanakan pengembangan dengan meningkatkan *reusabilitas* kode, serta memastikan konsistensi di seluruh aplikasi yang dikembangkan. Selain itu, pembaruan secara *real-time* milik *React Js* dapat memudahkan pengguna untuk memantau secara langsung [5]. Ditambah dengan *library Ant* Design mampu menghadirkan tampilan antarmuka yang lebih efisien dan menarik secara visual. Komponen *React Js* menjadi sangat penting dalam menciptakan *dashboard administrative* yang fungsional serta memiliki tampilan profesional dan *user-friendly*.

React Js yang dikombinasikan dengan *database NoSQL* seperti *MongoDB* mampu memberikan fleksibilitas dalam penyimpanan dan pengelolaan data. Kombinasi ini memungkinkan developer untuk menangani data yang tidak terstruktur dengan lebih efisien, sekaligus memanfaatkan fleksibilitas arsitektur *React Js* dalam pengembangan aplikasi. Selain itu, *React Js* dikenal karena kemampuan *framework*-nya yang adaptif, fleksibel, serta memiliki tingkat keamanan dan kinerja yang optimal [6]. Dengan demikian, penggunaan *React Js* dalam pengembangan sistem dengan basis *website* tidak hanya meningkatkan performa dan fungsi dari aplikasi, tetapi juga meningkatkan tingkat keamanan yang lebih tinggi, dan memberikan keunggulan bagi perusahaan yang menggunakannya.

2. Metodologi Penelitian

Pada penelitian yang berjudul "Perancangan Sistem *E-reporting* Menggunakan *React Js* dan *Firebase*" disebutkan bahwa dalam konteks pengembangan aplikasi *E-reporting*, teknologi *React Js* dan *Firebase* digunakan sebagai fondasi utama. Teknologi *React Js* digunakan untuk mengembangkan antarmuka pengguna yang responsif dan interaktif,

sementara *Firebase* digunakan sebagai platform *back end* untuk menyimpan data dan mengelola autentikasi pengguna. Dengan demikian, penggunaan kedua teknologi ini memberikan kemudahan dalam mengembangkan sistem pelaporan elektronik yang efisien dan dapat diakses secara *online* oleh para karyawan [7].

Penelitian "Perancangan Sistem *E-reporting* Menggunakan *React Js* dan *Firebase*" membahas pengembangan aplikasi E-reporting yang memungkinkan pembuatan dan akses laporan secara *online*, memudahkan karyawan dalam menyampaikan dan mengelola laporan. Penelitian saya, yang berfokus pada pengembangan *dashboard* admin menggunakan *React Js* dan *Ant Design*, juga berfungsi untuk menyediakan platform yang efisien bagi administrator dalam memantau dan mengelola laporan. Kedua penelitian ini memiliki hubungan erat dalam hal pengembangan sistem yang memungkinkan pembuatan, akses, dan pengelolaan laporan, meskipun difokuskan pada konteks yang berbeda, yaitu *E-reporting* dan *dashboard* admin.

Penelitian terdahulu sudah menggunakan *React Js* meskipun belum menggunakan *library Ant Design* milik *React*, sehingga pada penelitian ini membahas mengenai kelebihan dari *Ant Design* dalam segi *User Interface* (UI).

Pada penelitian yang berjudul "Pengembangan *Dashboard Trivy* Berbasis *Website* Menggunakan *React JS* dan *Golang*" disebutkan bahwa *React JS* mempermudah pengembangan aplikasi *website* karena didasarkan pada komponen yang interaktif, *stateful*, dan dapat digunakan kembali. Dengan menerapkan teknik *Single Page Application* (SPA), *React Js* dapat *merender* halaman lebih cepat karena hanya memuat data yang diperlukan oleh pengguna [8].

Kedua penelitian membahas tentang pengembangan *dashboard* menggunakan *React Js*. Meskipun penelitian pertama berfokus pada keamanan aplikasi dengan memindai kerentanan *container image* menggunakan *Trivy*, sementara penelitian yang kedua lebih menekankan pada pengembangan *dashboard* admin untuk toko dengan menggunakan *React Js*. Namun, keduanya memiliki kesamaan dalam mengakui kemudahan penggunaan *React Js* dalam membangun antarmuka pengguna yang interaktif, *stateful*, dan dapat digunakan kembali. Selain itu, kedua penelitian tersebut menyoroti kemampuan *React Js* dalam membangun aplikasi *Single Page Application* (SPA) yang efisien, di mana navigasi antar halaman dilakukan tanpa perlu memuat ulang halaman secara keseluruhan, sehingga meningkatkan pengalaman pengguna secara signifikan.

Penelitian terdahulu sudah menggunakan teknik *Single Page Application* (SPA) yang memungkinkan proses *rendering* halaman yang lebih cepat dengan hanya memuat data yang diperlukan oleh pengguna, sehingga meningkatkan pengalaman pengguna secara keseluruhan.

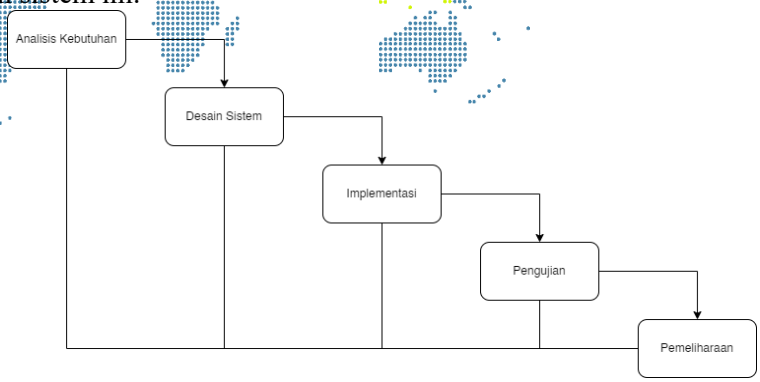
Pada penelitian yang berjudul "Sensor Network Simulator *Prototype With Real-Time Environmental* Data *Monitoring to Build Smart Application*" disebutkan bahwa aplikasi dibuat menggunakan *React Js* dan *Ant Design Framework* [9].

Di penelitian di atas mengembangkan sebuah *user web application* menggunakan *framework React Js* dan *Ant Design* dikarenakan *React* dan *Ant Design* menawarkan tampilan yang reaktif, responsif, dan pemuatan halaman yang cepat serta mudah untuk dimodifikasi ataupun dimanipulasi. Begitu juga di penelitian ini yang membahas mengenai pengembangan *dashboard* admin menggunakan *React* dan *Ant Design* dikarenakan banyak tampilan yang mudah untuk dimodifikasi dari *Ant Design* itu sendiri.

Ant Design Framework adalah salah satu *library* milik *React Js* yang berfokus pada *User Interface* atau tampilan, *Ant Design* menawarkan berbagai macam tampilan yang mudah untuk digunakan dan dimodifikasi sesuai dengan kebutuhan suatu *website*. Tampilan dari *Ant Design* juga responsif sehingga mampu menyesuaikan berbagai macam ukuran.

Dalam penelitian ini, metode yang digunakan dalam proses pengembangan sistem adalah *waterfall*. Model ini memiliki karakteristik yang terstruktur dan sistematis, sehingga memastikan bahwa pengembangan sistem berjalan secara teratur melalui

tahapan-tahapan yang jelas. Setiap tahap memiliki peran yang penting dalam memastikan semua aspek yang dibutuhkan oleh sistem terpenuhi sebelum masuk ke tahap berikutnya. Gambar 1 di bawah ini akan menggambarkan alur penelitian yang diikuti dalam proses pengembangan sistem ini.



Gambar 1. Alur penelitian

Model diagram *waterfall* adalah salah satu model yang paling sering diterapkan dalam pembuatan sistem. Berdasarkan pandangan Sommerville, model ini bersifat linier, dimulai dari tahap awal pengembangan sistem, yaitu perencanaan, hingga tahap akhir pengembangan, yakni pemeliharaan. Setiap tahap harus diselesaikan sepenuhnya sebelum tahap berikutnya dimulai, dan tidak ada kemungkinan untuk kembali atau mengulangi tahap sebelumnya [10].

Pada tahap analisis kebutuhan, dilakukan pengumpulan dan analisis menyeluruh terhadap kebutuhan sistem. Kegiatan ini melibatkan identifikasi kebutuhan pengguna, dalam hal ini administrator Toko Berkat, serta menentukan fitur-fitur utama yang akan ada pada *dashboard admin*. Fitur-fitur ini mencakup pemantauan data penjualan, manajemen produk, dan pelaporan. Selain itu, tahap ini juga mencakup penentuan kebutuhan teknis, termasuk pemilihan teknologi yang akan digunakan, yaitu *React JS*. *React*, atau yang juga dikenal sebagai *React.js* atau *React Js*, adalah *library front-end* berbasis *JavaScript* yang digunakan untuk membangun antarmuka pengguna atau komponen *User Interface (UI)*. *React* dikelola oleh Facebook serta komunitas pengembang dan perusahaan [11]. Lalu *React JS* digunakan juga bersama dengan *library Ant Design*. *Ant Design*, atau sering disebut *AntD*, adalah sistem desain yang dikembangkan oleh Alibaba *Group*. Sistem ini menyediakan *library* komponen untuk berbagai *framework JavaScript*, seperti *React* [12].

Tahap desain sistem melibatkan perancangan arsitektur dan antarmuka pengguna dari *dashboard admin*. Kegiatan ini meliputi pembuatan desain arsitektur aplikasi, yang mencakup struktur komponen *React* dan integrasi dengan *Ant Design*. *Ant Design* (*Antd*), sebuah *library* antarmuka pengguna, digunakan untuk merancang proyek penjadwalan janji temu dokter ini. *Library* ini membantu pengembang dengan cepat membuat antarmuka web yang menarik dan fungsional. *Ant Design* menawarkan berbagai komponen yang tersedia dan mengikuti bahasa desain yang konsisten, sehingga membuat pengembangan menjadi efisien dan mudah (*Ant Design*, n.d) [13]. Desain tampilan antarmuka pengguna (*UI*) dibuat menggunakan *Ant Design* untuk memastikan responsivitas dan kemudahan penggunaan. Selain itu, *mockup* dan *prototype* awal dibuat untuk mendapatkan umpan balik dari pengguna, sehingga dapat dilakukan penyesuaian sebelum tahap implementasi.

Pada tahap implementasi, desain yang telah dibuat diimplementasikan menjadi kode program. Kegiatan ini melibatkan pengembangan komponen-komponen *React* sesuai dengan desain yang telah dibuat, serta integrasi *Ant Design* untuk membangun antarmuka pengguna yang interaktif dan responsif. Pengujian unit dilakukan untuk memastikan

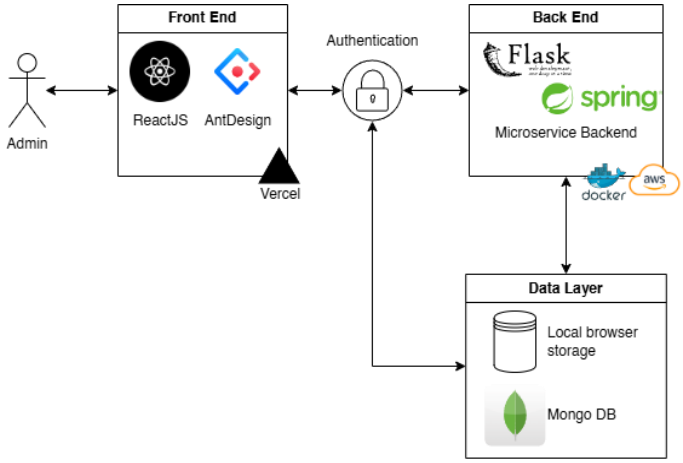
setiap komponen berfungsi dengan baik, sehingga dapat mengurangi kemungkinan kesalahan pada tahap pengujian selanjutnya.

Tahap pengujian bertujuan untuk memastikan bahwa *dashboard admin* berfungsi sesuai dengan kebutuhan yang telah ditentukan. Kegiatan ini meliputi pengujian integrasi untuk memastikan bahwa semua komponen bekerja bersama dengan baik, serta pengujian fungsional untuk memastikan bahwa semua fitur berfungsi sesuai dengan spesifikasi. Selain itu, dilakukan pengujian penerimaan pengguna (*User Acceptance Testing*) dengan melibatkan pengguna akhir, yaitu administrator Toko Berkat, untuk memastikan bahwa sistem telah memenuhi kebutuhan dan harapan mereka.

Setelah sistem diimplementasikan dan diuji, tahap pemeliharaan dilakukan untuk memastikan sistem tetap berjalan dengan baik dan sesuai dengan kebutuhan. Kegiatan ini meliputi pemantauan kinerja sistem dan perbaikan bug yang ditemukan, melakukan update dan penambahan fitur sesuai dengan kebutuhan pengguna, serta memberikan dukungan teknis dan pelatihan kepada pengguna. Tahap ini penting untuk memastikan bahwa sistem dapat terus digunakan dengan optimal dan dapat beradaptasi dengan perubahan kebutuhan di masa depan.

3. Hasil dan Pembahasan

Dalam pengembangan aplikasi *dashboard administrative* untuk CV Berkat ini berbagai tahapan metodologi penelitian telah diterapkan untuk memastikan hasil yang optimal dan sesuai dengan kebutuhan pengguna. Proyek ini dirancang dengan menggunakan *framework React Js* dan *library*-nya yaitu *Ant Design* untuk tampilan antarmukanya. Dengan menerapkan model *waterfall*, setiap tahap pengembangan, mulai dari analisis kebutuhan hingga tahap pemeliharaan dapat berjalan secara sistematik dan menghasilkan aplikasi berbasis *website* yang sesuai dengan ekspektasi pengguna.



Gambar 2. Arsitektur Sistem

Alur arsitektur sistem pada aplikasi ini dimulai dari *front end* yang dikembangkan menggunakan *React Js*, dimana pengguna akan berinteraksi dengan tampilan antarmuka aplikasi *web.* Ketika mengakses *dashboard* atau melakukan *request* data, *React Js* akan mengirimkan *HTTP Request* ke *service back end* melalui *API. API* ini akan berperan sebagai perantara yang memproses permintaan serta menjalankan logika dan berinteraksi dengan *data layer* untuk mengambil atau melakukan manipulasi data sesuai dengan kebutuhan.

Di *framework React Js* penggunaan *hooks* ditujukan untuk mengelola *state* dan efek. Dengan adanya *hooks* kode menjadi lebih ringkas dan modular. Di aplikasi *dashboard administrative* ini digunakan dua *hooks* yaitu *useState* dan *useEffect*.

a) Kode program 1 *useState* dalam *Dashboard.js*

```
const [inventory, setInventory] = useState([]);
const [employees, setEmployees] = useState([]);
const [transactions, setTransactions] = useState([]);
const [startDate, setStartDate] = useState(null);
const [endDate, setEndDate] = useState(null);
```

UseState adalah salah satu *hook* yang ada pada *React Js* dan digunakan untuk mengelola *state* lokal dalam suatu komponen. *State* dalam *react* merujuk kepada data atau status yang dimiliki oleh suatu komponen dan dapat dirubah selama siklus komponen tersebut hidup. Ketika terjadi perubahan pada *state*, *react* akan secara otomatis merender ulang komponen agar mencerminkan *state* terbaru. *UseState* sendiri mengembalikan dua nilai elemen yaitu nilai *state* saat ini dan fungsi untuk memperbarui nilai *state* tersebut.

b) Kode program 2 *useEffect* dalam *Dashboard.js*

```
useEffect(() => {
    if (token) {
      axios
        .get(`${config.API_BASE_URL}/home`, {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        })
        .then((response) => {
          setInventory(response.data.inventory_summary);
          setEmployees(response.data.employee_summary);
        })
        .catch((error) => {
          console.error("Error:", error.response);
          if (error.response.status === 401) {
            navigate("/");
          }
        });
    } else {
      navigate("/");
      return;
    }
  }, [navigate, token]);
```

Lalu untuk *hook* selanjutnya adalah *useEffect, useEffect* memungkinkan pengelolaan efek dalam komponen lebih fungsional. Efek yang dimaksud mencakup proses-proses di luar siklus utama *render* komponen, seperti pengambilan data dari API (*fetching* API) dan manipulasi DOM. *UseEffect* memberikan kontrol atas kapan dijalankannya fungsi tersebut setelah terjadi render, memastikan bahwa komponen melakukan pembaruan ketika suatu kondisi terpenuhi.

React router digunakan untuk mengatur navigasi antar halaman yang ada di aplikasi ini. Setiap rute dihubungkan dengan komponen yang sesuai agar data yang ditampilkan sesuai atau relevan.

Context API adalah fitur yang efektif dan memudahkan proses *maintenance* atau pemeliharaan, serta dengan *context API* juga akan lebih mudah dalam memahami alir data [14]. Di dalam aplikasi ini, *Context API* digunakan untuk mengatur dan mengelola *state global* yang kemudian dapat diakses oleh beberapa komponen. Tujuan dipakainya metode ini adalah untuk menghindari pengiriman *props* secara berlebihan di sepanjang kode komponen. *Context* API memungkinkan untuk membuat variabel yang digunakan untuk menyimpan data global dan memberikan izin komponen lain untuk mengaksesnya dengan mudah.

c) Kode program 3 *config.js*

```
const config = {
   API_BASE_URL: "https://api.jinzyy.site", //Docker
   //API_BASE_URL: "http://localhost:5000", //Local
};

export default config;
```

Akun dengan otoritas atau *role* sebagai *admin* adalah satu-satunya yang dapat melakukan *login* ke sistem, untuk akun lain dengan otoritas selain *admin* seperti pegawai tidak akan bisa mengakses sistem. Pembatasan ini dirancang untuk menjaga keamanan serta kerahasiaan data penting yang hanya boleh diakses oleh *admin*. Aplikasi ini menyediakan fitur untuk melihat pendapatan harian berdasarkan rentang tanggal yang dipilih, dengan data yang disajikan dalam bentuk grafik. Selain itu *admin* juga dapat melakukan operasi *Create, Read, Update, Delete* (CRUD) pada daftar barang dagangan dan pengelolaan karyawan

Fitur lain dari aplikasi ini adalah *admin* dapat meninjau laporan transaksi sesuai dengan rentang tanggal dan data dapat diekspor ke dalam format *excel* atau CSV. Dengan opsi rentang tanggal, akan sangat berguna untuk membuat laporan bulanan dan CSV sebagai *backup* data saat ingin melakukan *maintenance* rutin. Aplikasi ini juga memungkinkan untuk melakukan *import* data dari CSV, sehingga memudahkan untuk mengisi ulang data berdasarkan data yang telah di *backup* atau bahkan migrasi ke *database* yang baru.

Aplikasi ini di *deploy* menggunakan server dari AWS EC2, dengan sistem operasi ubuntu 22.04.05, serta memanfaatkan beberapa teknologi agar *back end* dan *front end* berjalan dengan baik. *Back end* aplikasi ini menggunakan *framework* dari *python* yaitu *flask* yang sudah berjalan pada *docker container*, sementara *front end* menggunakan *vercel* sebagai media *hosting*-nya.

Pengujian performa *React Js* ini dilakukan menggunakan *Lighthouse*, *Lighthouse* sendiri adalah ekstensi dari *web browser* yang digunakan untuk menguji halaman *website* dan *progressive web apps* [15]. Pengujian terhadap *website* ini dilakukan terhadap empat poin utama, yaitu *performance, accessibility, best practices,* dan *Search Engine Optimization* (*SEO*). Hasil dari pengujian *website* ini disajikan dalam **tabel 1** dan disertai dengan detail matriks yang penting yang ditunjukkan pada **tabel 2** berdasarkan *lighthouse*,

Tabel 1. Pengujian Lighthouse

Aspek	***Hosting***	Lokal
Performance	90	55
Accessibility	97	97
Best Practices	96	93
Search Engine Optimization (SEO)	91	91

Tabel 2. Detail Matriks

Aspek	***Hosting***	Lokal
First Contentful Paint	0,4 detik	0,9 detik
Largest Contentful Paint	0,8 detik	3,5 detik
Total Blocking Time	60 ms	280 ms
Speed Index	1,0 detik	2,4 detik
Cumulative Layout Shift	0,197	0,197

Pada grafik pengujian *lighthouse,* dapat terlihat bahwa skor performa aplikasi di lingkungan *hosting* jauh lebih tinggi daripada di lokal. Skor performa di *hosting* mencapai sekitar 90, sementara di lokal hanya di angka 55. Hal ini mengindikasikan bahwa aplikasi dapat berjalan lebih cepat saat di hosting. Faktor ini dapat diakibatkan oleh penggunaan server yang lebih baik dan *caching* yang lebih efisien di lingkungan hosting.

Pada aspek aksesibilitas, *best practices,* dan SEO, skor yang diperoleh hampir sama antara *hosting* dan lokal. Hal ini menunjukkan bahwa konfigurasi elemen, *best practices,* dan optimalisasi SEO pada kedua lingkungan sudah diimplementasikan dengan baik. Skor pada ketiga aspek ini menandakan bahwa aplikasi telah memperhatikan kemudahan akses pengguna.

Di *hosting,* FCP berada di angka 0,4 detik, sedangkan di lokal mendapat angka 0,9 detik. FCP mengukur waktu yang dibutuhkan untuk menampilkan elemen pertama yang dimuat pada halaman. Waktu FCP yang lebih rendah di *hosting* menunjukkan bahwa aplikasi dapat menampilkan konten pertama dengan lebih cepat, yang membuat pengguna tidak perlu menunggu lama untuk melihat tampilan halaman.

LCP di *hosting* adalah 0,8 detik sementara di lokal membutuhkan 3,5 detik. LCP adalah waktu yang dibutuhkan untuk menampilkan elemen terbesar, seperti gambar atau elemen utama. Perbedaan yang signifikan ini menunjukkan bahwa di lingkungan *hosting*, elemen utama dapat dimuat lebih cepat dibandingkan dengan lokal.

TBT pada *hosting* mendapat kecepatan 60 ms, sementara di lokal mendapat 280 ms. TBT mengukur total waktu ketika halaman tidak dapat berinteraksi karena proses yang memblokir seperti skrip berat. Skor TBT di bawah 100 ms menunjukkan bahwa pengguna dapat berinteraksi lebih cepat dengan halaman.

Di *hosting,* skor *speed index* mendapat waktu 1,0 detik, sementara di lokal 2,4 detik. *Speed index* menunjukkan seberapa cepat konten halaman dapat terlihat sepenuhnya oleh pengguna. Nilai yang lebih rendah di *hosting* menunjukkan bahwa seluruh halaman bisa terlihat lebih cepat, sehingga pengguna dapat dengan segera melihat dan menggunakan aplikasi.

Nilai CLS pada kedua lingkungan baik *hosting* maupun lokal mendapat skor yang sama yaitu 0,197, menunjukkan bahwa tidak ada pergeseran tata letak yang signifikan pada saat memuat halaman. CLS adalah nilai matriks yang mengukur visual dalam hal stabilitas, dimana tata letak halaman tidak mengalami perubahan atau pergeseran pada saat konten dimuat. Nilai CLS yang baik ini juga menandakan bahwa pengguna tidak akan terganggu oleh perubahan tata letak yang berubah-ubah pada saat memuat halaman.

Dari data di atas, performa aplikasi jauh lebih optimal saat berjalan di lingkungan *hosting* dibandingkan dengan lokal. Hal ini dapat terlihat pada matriks LCP, TBT, dan *speed indeks* yang memiliki perbedaan yang signifikan. Hal ini dapat diakibatkan oleh faktor-faktor lain seperti, server yang lebih optimal, penggunaan *Content Delivery Network* (CDN), serta *caching* yang lebih baik.

4. Kesimpulan

Berdasarkan hasil pengembangan dan implementasi menggunakan *React Js* pada *dashboard administrative* ini dapat disimpulkan bahwa penggunaan *dashboard* yang menggunakan *framework react* ini memberikan kemudahan dalam memantau data penjualan, karyawan, dan inventaris toko di CV Berkat. Dengan memanfaatkan tampilan antarmuka yang interaktif aplikasi ini mampu menyajikan informasi yang dibutuhkan secara *real-time*. *Framework* ini memberikan pengelolaan *state* yang lebih baik dengan menggunakan *useState* dan *useEffect*. Selain itu kombinasinya dengan *library Ant Design* mempercepat proses pengembangan aplikasi dengan berbagai komponen yang sudah tersedia. *React Js* dalam aplikasi ini menggunakan metode *Single Page Application* (SPA) yang memungkinkan aplikasi untuk merender halaman dengan lebih cepat tanpa perlu dimuat ulang secara penuh, memberikan pengalaman yang lebih baik dari sisi user juga. Sistem ini juga mudah untuk melakukan perubahan atau modifikasi di masa mendatang beriringan dengan kebutuhan *user*. Secara keseluruhan, *dashboard* yang dikembangkan dengan *React Js* dan *Ant Design* telah berhasil memenuhi kebutuhan di CV Berkat dalam segi efisiensi operasionalnya. Teknologi ini juga memberikan penawaran terhadap pengembangan fitur-fitur baru di masa mendatang, sesuai dengan kebutuhan bisnis yang terus berkembang.

Daftar Pustaka

[1] A. R. Akhdani, "Implementasi React.Js Pada Pengembangan Frontend Sistem Informasi Manajemen Kader Partai," Universitas Islam Indonesia, Yogyakarta, 2022.

[2] M. Ramadhan, A. M. Thantawi, dan S. Setiawati, "Rancang Bangun Dashboard Admin Monitoring Pencapaian Prestasi Belajar Siswa Di SMKN 33 Jakarta Berbasis Web," *Open Journal System Universitas Persada Indonesia Y.A.I*, vol. 7, hlm. 72, Jul 2023, [Daring]. Tersedia pada: https://journals.upi-yai.ac.id/index.php/ikraith-informatika/issue/archive

[3] A. Azizi, M. Naufal, A. Ghazali, O. S. Khair, Z. Tsabit, dan S. Kusnandar, "Pengembangan Dashboard Admin Bukupedia," *Jurnal Teknik Indonesia*, vol. 3, hlm. 13–21, Jan 2024, doi: 10.58860/jti.v3i1.

[4] N. David Maria Veronika, Y. Darnita, U. Muhammadiyah Bengkulu, dan I. Korespondensi, "Perancangan Sistem Dashboard Monitoring Data Pelanggan Perusahaan Air Minum Daerah (Perumda) Tirta Hidayah Kota Bengkulu Berbasis Website," 2022.

[5] G. Sai Teja, D. Deepak Manchala, G. Bhargav, dan M. K. Naga Sailaja, "Empowering Enterprise Development by Building and deploying Admin Dashboard using Refine Framework," *Cornell University*, hlm. 1–12, Apr 2024.

[6] F. T. Anaclaudia, D. Pramana, dan I. M. B. Saputra, "Reactjs and Expressjs Implementation In PMK ITB STIKOM Bali Activity Management," *APTISI Transactions on Technopreneurship*, vol. 5, no. 3, hlm. 216–226, Nov 2023, doi: 10.34306/att.v5i3.313.

[7] J. Panjaitan dan A. F. Pakpahan, "Perancangan Sistem E-Reporting Menggunakan ReactJS dan Firebase," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 1, Apr 2021, doi: 10.28932/jutisi.v7i1.3098.

[8] A. A. Kroons dan C. Dewi, "Pengembangan Dashboard Trivy Berbasis Website Menggunakan React Js Dan Golang," *Jurnal Indonesia : Manajemen Informatika dan Komunikasi*, vol. 4, no. 3, hlm. 1037–1049, Sep 2023, doi: 10.35870/jimik.v4i3.295.

[9] W. Velasquez, "Sensor Network Simulator Prototype With Real-Time Environmental Data Monitoring to Build Smart Application," *IEEE Access*, vol. 9, hlm. 144530–144539, 2021, doi: 10.1109/ACCESS.2021.3122829.

[10] V. Adi Kurniyanti dan D. Murdiani, "Perbandingan Model Waterfall Dengan Prototype Pada Pengembangan System Informasi Berbasis Website," *Jurnal Syntax Fusion*, vol. 2, no. 08, hlm. 669–675, Agu 2022, doi: 10.54543/fusion.v2i08.210.

[11] M. Fariz, S. Lazuardy, dan D. Anggraini, "Modern Front End Web Architectures with React.Js and Next.Js," *International Research Journal of Advanced Engineering and Science*, vol. 7, no. 1, hlm. 132–141, 2022.

[12] S. Salonen, "Evaluation Of Ui Component Libraries In React Development," *Tampereen yliopisto*, hlm. 1–46, Apr 2023.

[13] N. Tran, "Application for booking doctor ap-pointments Implement and Design a Full-Stack Booking Doctor Appointment," *Tampere University of Applied Sciences*, hlm. 1–48, Mei 2024.

[14] T. Le, "Comparison of State Management Solutions between Context API and Redux Hook in ReactJS Title: Comparison of State Management Solutions between Context API and Redux Hook in ReactJS," *Metropolia*, hlm. 1–42, Mar 2021.

[15] M. J. Dewi dan Nurdin, "Analisis Performa Platform Sosial Media Menggunakan Perbandingan Software Automated Testing," *Information Management for Educators and Professionals*, vol. 7, no. 2, hlm. 164–173, Jun 2023.