TEKNO Jurnal Teknologi Elektro dan Kejuruan

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

Implementasi *Single Page Application* Pada Sistem Pemesanan Multi-Layanan Travel Berbasis Web Menggunakan Next.Js dan *Typescript*

Adelia Miftakul Janah[1], Harits Ar Rosyid[2]

1. Universitas Negeri Malang, Indonesia | adelia.miftakul.2205356@students.um.ac.id
2. Universitas Negeri Malang, Indonesia | harits.ar.ft@um.ac.id

Abstrak

Perkembangan teknologi informasi yang pesat mendorong kebutuhan akan platform digital yang responsif dan terintegrasi. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi sistem pemesanan travel multi-layanan menggunakan pendekatan *Single Page Application* (SPA) dengan Next.js dan *TypeScript*. Proses pengembangan meliputi analisis kebutuhan, perancangan sistem, implementasi, serta pengujian performa menggunakan *Google Lighthouse* berdasarkan metrik *Core Web Vitals*. Sistem SPA ini di-*hosting* di Vercel dan diuji melalui indikator performa seperti *Largest Contentful Paint* (LCP), *Cumulative Layout Shift* (CLS), *Speed Index* (SI), *First Contentful Paint* (FCP), dan *Total Blocking Time* (TBT). Hasil pengujian menunjukkan bahwa aplikasi SPA yang dikembangkan memperoleh skor kategori baik di seluruh metrik, dengan rata-rata LCP 0,5; CLS 0,0015; SI 0,6375; FCP 0,425; dan TBT 40. Temuan ini menunjukkan bahwa integrasi Next.js dan *TypeScript* secara signifikan dapat meningkatkan performa, menjaga stabilitas kode, serta memberikan pengalaman pengguna yang unggul pada aplikasi web.

Kata Kunci

Single Page Application; Next.js; *TypeScript*; *Core Web Vitals*; pengujian performa; aplikasi web

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

1. Pendahuluan

Perkembangan teknologi informasi dan komunikasi telah membawa perubahan besar dalam berbagai aspek kehidupan. Kemajuan teknologi informasi telah memengaruhi hampir semua aspek kehidupan manusia, dari pekerjaan hingga hiburan (Muttaqin *et al.*, 2021). Salah satu dampak nyata dari kemajuan ini terlihat dalam pengembangan aplikasi, khususnya website. Aplikasi berbasis web kini tidak lagi bersifat statis, melainkan interaktif dan responsif, sehingga mampu memberikan pengalaman pengguna yang lebih baik dan dinamis.

Di era digital saat ini, kebutuhan akan sistem pemesanan layanan travel yang terintegrasi, meliputi tiket transportasi, akomodasi, hingga paket wisata semakin meningkat. Hal ini diperkuat oleh data pada tahun 2023 yang menunjukkan bahwa sebanyak wisatawan 72% lebih memilih memesan perjalanan secara daring dibandingkan dengan hanya 12% yang masih mengandalkan agen perjalanan (Travel Perk, 2024). Selain itu, penggunaan aplikasi tumbuh sebesar 36% dari Januari 2020 hingga Desember 2021, sementara penggunaan situs web tumbuh hingga 57% (Amplitude Labs, 2022).

Tren ini menunjukkan adanya pergeseran perilaku konsumen menuju pemanfaatan platform digital yang lebih fleksibel dan cepat. Pengelolaan usaha travel memerlukan pelayanan konsumen yang cepat, sehingga usaha ini membutuhkan teknologi informasi web yang tepat untuk membantu meningkatkan kualitas pelayanannya (Fauziah *et al.*, 2019). Kondisi ini menuntut adanya solusi berbasis web yang tidak hanya mudah diakses, tetapi juga memiliki performa tinggi dan kemudahan penggunaan. Oleh sebab itu, paradigma pengembangan aplikasi berbasis *Single Page Application* (SPA) menjadi semakin populer karena kemampuannya dalam mengurangi waktu muat ulang halaman dan meningkatkan interaktivitas antar muka pengguna secara signifikan.

SPA merupakan aplikasi yang bekerja di dalam *browser* yang tidak membutuhkan *reload page* saat digunakan (Nariswari dan Kadyanan, 2023). Pendekatan pengembangan aplikasi web ini memungkinkan seluruh konten aplikasi dimuat dalam satu halaman, dengan pembaruan konten dilakukan secara dinamis tanpa perlu memuat ulang halaman secara penuh. Pendekatan ini sangat cocok untuk aplikasi dengan interaksi kompleks dan *real time*, seperti sistem pemesanan multi-layanan travel. Dalam membangun SPA yang efisien, *framework* modern seperti Next.js hadir sebagai solusi unggulan. NextJS memungkinkan komponen yang sama pada halaman lain tidak perlu dimuat ulang, hanya tampilan yang berubah yang diperbarui (Next.js, 2024). *Framework* ini juga menyediakan fitur *Client Side Rendering* (CSR) untuk meningkatkan interaktivitas website dengan kecepatan render yang lebih cepat dibandingkan *Server Side Rendering* (SSR) (Nextjs, 2024).

Seiring dengan itu, penggunaan *TypeScript* juga semakin meluas dalam pengembangan *frontend* yang modern. *TypeScript* merupakan sebuah *typed superset* dari *JavaScript* yang diciptakan untuk mengatasi tantangan yang ditimbulkan oleh sistem pengetikan dinamis *JavaScript* dan untuk meningkatkan kemampuannya. Dengan memperkenalkan sistem tipe yang

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

eksplisit, *TypeScript* membantu pengembang menemukan kesalahan sejak awal dalam proses pengembangan, sehingga mengurangi kemungkinan terjadinya kesalahan saat aplikasi dijalankan (*runtime errors*) (Kacper Rafalski, 2024). Fitur seperti ini sangat berguna dalam mengelola proyek besar dan kompleks, seperti sistem pemesanan multi-layanan travel yang terdiri dari berbagai modul dan layanan.

Di sisi lain, pentingnya standar pengukuran performa website juga menjadi aspek krusial dalam konteks pengembangan aplikasi web. Persepsi terhadap performa bisa berbeda-beda tergantung pada pengguna, sehingga diperlukan tolok ukur objektif untuk evaluasi. Salah satu standar yang diakui secara luas adalah *Core Web Vitals*. *Core Web Vitals* adalah metrik penting yang digunakan untuk mengukur aspek-aspek utama dari pengalaman pengguna di situs web, seperti kecepatan muat, interaktivitas, dan stabilitas visual (Anugerah Widi, Eko Sediyono, 2024). Untuk mengukur metrik tersebut, pengembang dapat menggunakan *Google Lighthouse*. *Google Lighthouse* sendiri adalah ekstensi *Google Chrome* yang digunakan untuk menguji halaman website dan *progressive web apps* (Aripin dan Somantri, 2021). Oleh karena itu, pengembangan SPA tidak hanya harus berfokus pada sisi teknis, tetapi juga pada pengukuran dan peningkatan performa secara objektif dan terukur.

Berdasarkan hal tersebut, penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi sistem pemesanan multi-layanan travel berbasis web dengan pendekatan SPA menggunakan Next.js dan *TypeScript*. Tujuan dari penelitian ini adalah untuk menganalisis sejauh mana penggunaan kedua teknologi tersebut dapat meningkatkan efisiensi pengembangan, stabilitas kode, dan kualitas pengalaman pengguna pada aplikasi travel berbasis web. Melalui studi ini, diharapkan diperoleh gambaran yang jelas mengenai manfaat praktis dari integrasi Next.js dan *TypeScript* dalam pengembangan aplikasi web modern yang kompleks dan multi-layanan.

2. Metode

Pelaksanaan tahap pengembangan ini dapat dijadikan sebagai metode yang efisien dalam membangun website. Proses ini juga berfungsi untuk memastikan bahwa seluruh aspek penelitian teridentifikasi dengan jelas serta sistem yang dikembangkan sesuai dengan kebutuhan pengguna.

1) Analisis Kebutuhan

Perancangan sistem website yang optimal dan memiliki performa yang baik memerlukan penggunaan alat-alat pendukung dalam proses pembuatannya. Salah satu langkah awal yang penting adalah melakukan analisis kebutuhan sistem. Kebutuhan sistem ini dibagi menjadi dua kategori utama, yaitu kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja/layanan apa saja yang nantinya harus disediakan oleh sistem, mencakup bagaimana sistem harus bereaksi pada input tertentu dan bagaimana perilaku sistem pada situasi tertentu. Sedangkan kebutuhan non fungsional adalah adalah

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

kebutuhan yang menitikberatkan pada properti prilaku yang dimiliki oleh sistem, kebutuhan fungsional juga sering disebut sebagai batasan layanan atau fungsi yang ditawarkan sistem seperti batasan waktu, batasan pengembangan proses, standarisasi dan lain lain (Kherina Surya Ningsih, Nur Jamilah Aruan, 2022). Tabel 1 menyajikan hasil analisis kebutuhan untuk sistem yang akan dikembangkan.

Tabel 1. Analisis Kebutuhan

Kebutuhan Fungsional	Kebutuhan Nonfungsional
Aplikasi wajib menampilkan data yang relevan dan sesuai dengan kebutuhan pengguna.	Aplikasi harus mampu memuat konten halaman secara dinamis tanpa perlu merefresh seluruh halaman.
Aplikasi wajib mengeksekusi perintah berdasarkan *input* pengguna, seperti proses transaksi atau pengecekan informasi.	Aplikasi harus mampu merespons perubahan konten pada halaman secara otomatis tanpa harus merefresh halaman.

Tabel 1. menjelaskan dua jenis kebutuhan utama yang menjadi dasar dalam pengembangan sistem. Kebutuhan fungsional menggambarkan apa saja yang harus dilakukan oleh sistem untuk memenuhi tujuan penggunaannya, misalnya menampilkan data dan menjalankan perintah yang dimasukkan oleh pengguna. Sedangkan kebutuhan nonfungsional lebih mengarah pada karakteristik performa sistem, seperti kecepatan dalam memuat konten dan kemampuan untuk memperbarui halaman secara dinamis tanpa perlu proses pemuatan ulang (*refresh*) secara keseluruhan.

Pemahaman terhadap kebutuhan-kebutuhan ini sangat penting karena akan memengaruhi bagaimana sistem dirancang dan diimplementasikan. Dengan mengetahui proses-proses penting yang harus dilakukan sistem serta batasan dan standar teknis yang harus dipenuhi, pengembang dapat menentukan arsitektur dan alur kerja sistem secara lebih tepat.

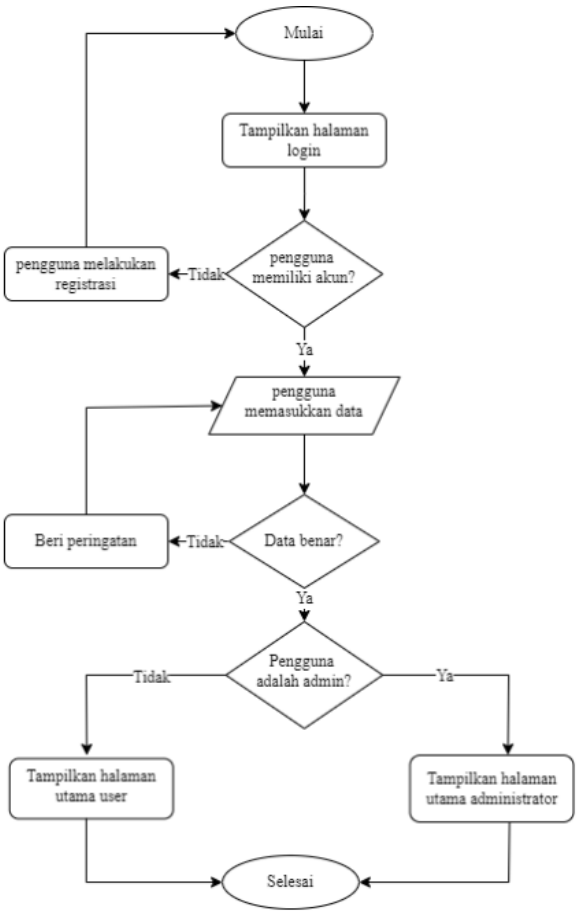
2) Perancangan Sistem

a) Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutanurutan prosedur dari suatu program (Zalukhu *et al.*, 2023). Dalam konteks pengembangan aplikasi, *flowchart* disusun sebagai visualisasi alur kerja yang memberikan gambaran terstruktur mengenai cara aplikasi berfungsi. *Flowchart* ini berperan penting dalam memfasilitasi proses pengembangan, analisis, dan pemeliharaan aplikasi secara lebih efisien. Gambar 1 dan 2 berikut adalah beberapa *Flowchart* yang menggambarkan proses utama pada sistem pemesanan travel. Gambar 1. memperlihatkan *flowchart* yang memvisualisasikan proses *login* pengguna dalam sistem. Sedangkan Gambar 2. adalah *flowchart* yang menggambarkan proses pencarian dalam sistem. *Flowchart* ini mengasumsikan bahwa pengguna sudah memiliki akun dan telah terdaftar sebagai pengguna layanan travel.

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

Setelah memilih salah satu layanan di halaman utama, pengguna akan diarahkan ke halaman daftar layanan tersebut seperti pada Gambar 4. Di halaman ini, dapat melihat berbagai pilihan dari layanan yang diinginkan yang ditampilkan secara dinamis

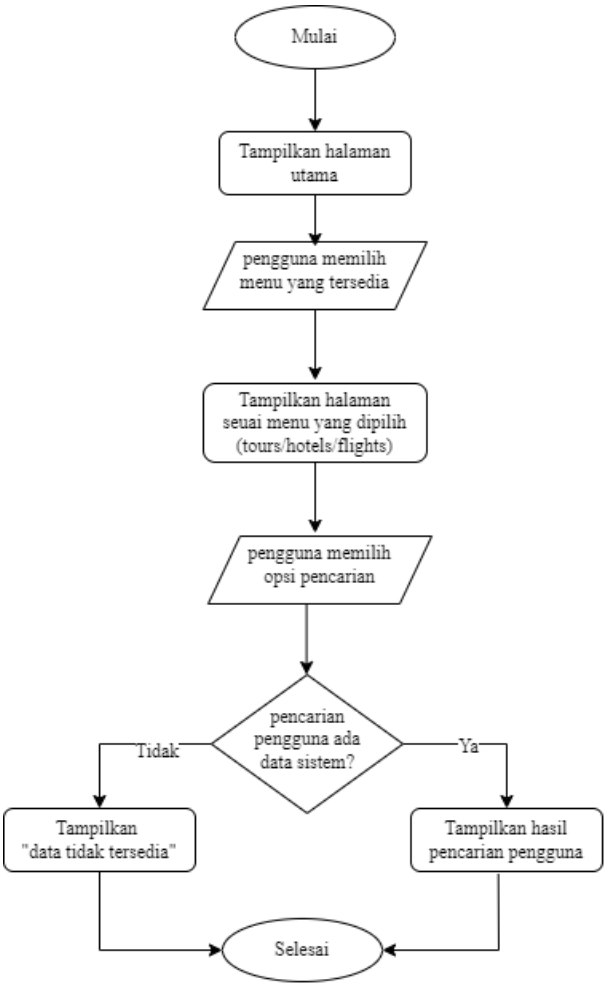


Gambar 1. *Flowchart* Login

b. User Interface (UI) *Design*

User Interface (UI) adalah antarmuka pengguna merupakan bentuk tampilan grafis yang berhubungan langsung dengan pengguna. Antarmuka pengguna berfungsi untuk menghubungkan antara pengguna dengan sistem operasi, sehingga komputer tersebut bisa digunakan (Djauhari *et al.*, 2023). Tahapan ini merupakan proses perancangan *mockup* desain antarmuka yang akan dijadikan acuan dalam pengembangan tampilan aplikasi. *Mockup* ini dirancang menggunakan aplikasi Figma. Gambar 3. menampilkan desain halaman utama dari aplikasi berbasis SPA. Halaman ini menyajikan berbagai layanan travel yang tersedia. Pengguna dapat memilih layanan yang diinginkan tanpa perlu memuat ulang halaman secara keseluruhan.

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657



Gambar 2. *Flowchart* Pencarian

3) Implementasi

Tahap implementasi merupakan proses merealisasikan seluruh rancangan yang telah disusun sebelumnya ke dalam bentuk sistem yang dapat dijalankan.

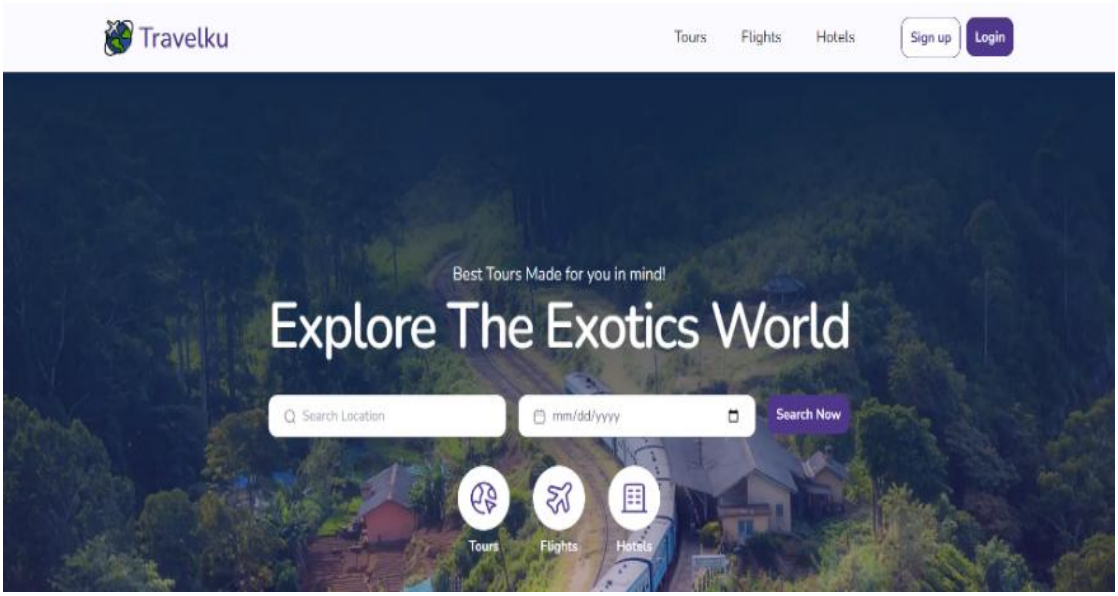
4) Pengujian

a. Parameter dan Metrik Pengujian

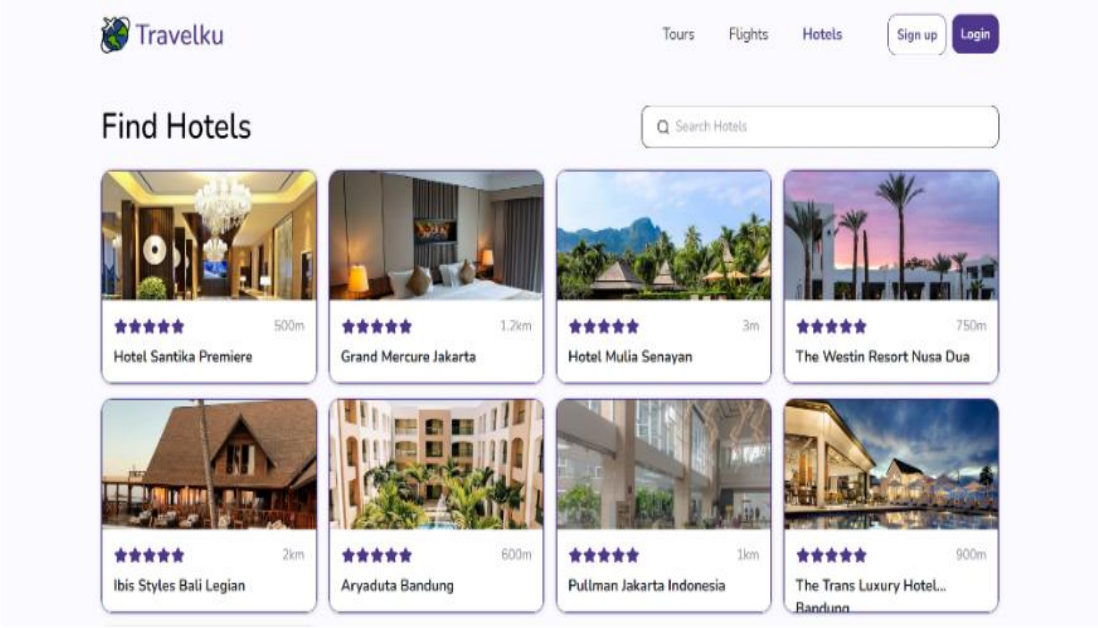
Pengujian pada sistem pemesanan multi-layanan travel berbasis web ini dilakukan dengan menggunakan berbagai parameter dan metrik performa yang sudah diakui secara luas. Metrik tersebut berfokus pada aspek kecepatan *loading*, interaktivitas, dan stabilitas visual agar

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

pengalaman pengguna menjadi optimal. Penggunaan metrik ini penting agar aplikasi dapat dinilai secara objektif dan sesuai dengan standar industri, seperti *Core Web Vitals* dari *Google*.



Gambar 3. Halaman Utama



Gambar 4. Halaman Menu Tiap Layanan

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

Pengujian pada sistem pemesanan multi-layanan travel berbasis web ini dilakukan dengan menggunakan beberapa metrik utama, yaitu *Largest Contentful Paint* (LCP), dan *Cumulative Layout Shift* (CLS). Selain itu, pengujian juga melibatkan metrik performa lainnya seperti *Speed Index* (SI), *First Contentful Paint* (FCP), dan *Total Blocking Time* (TBT).

b) Keterangan Parameter Pengujian

Tabel 2 memperlihatkan parameter pengujian yang dilakukan.

Tabel 2. Parameter Pengujian

Parameter Pengujian	Keterangan
LCP	Metrik yang menunjukkan waktu *loading* website, yang menilai seberapa cepat halaman web merender elemen terbesarnya, termasuk blok gambar dan teks.
CLS	Metrik yang menilai stabilitas visual dan mengecek apakah ada pergeseran *layout* yang tidak terduga di halaman.
SI	Metrik yang mengukur berapa lama waktu yang diperlukan untuk menampilkan bagian-bagian dalam *viewport*. Untuk menghitung SI, *tool* analisis akan merekam video saat halaman dimuat di *browser* dan menghitung prosesnya per-*frame*.
FCP	Metrik yang menghitung berapa lama *browser* menampilkan konten dari *Document Object Model* (DOM)
TBT	Total waktu halaman diblokir yang membuatnya tidak bisa merespon *input* pengguna.

Sumber: (Faradilla Ayunindya, 2025)

c. Interpretasi Skor Pengujian

Setiap skor hasil pengujian akan diinterpretasikan berdasarkan standar yang ditetapkan oleh *Google Lighthouse*, yaitu *Core Web Vitals*. Tabel 3 merupakan tolak ukur dari masing-masing parameter yang telah ditentukan oleh *Core Web Vitals*.

Tabel 3. Tabel Interpretasi Skor

Parameter Pengujian	Rentang Waktu Berdasarkan Kategori		
	Hijau (baik)	Oranye (perlu peningkatan)	Merah (buruk)
LCP	< 1.8	1.8 – 3.0	> 3,0
CLS	< 0,1	0,1 – 0,25	> 0,25
SI	0 – 1,3	1,3 – 2,3	> 2,3
FCP	0 – 1,8	1,8 – 3	> 3
TBT	0 – 150	150 – 350	> 350

Largest Contentful Paint (LCP)

Berdasarkan tabel interpretasi skor LCP, LCP yang baik adalah ketika waktu pemuatan elemen konten terbesar pada halaman terjadi kurang dari 1,8 detik. Hal ini menunjukkan bahwa halaman dapat menampilkan konten utamanya dengan sangat cepat sehingga memberikan pengalaman pengguna yang optimal. LCP yang memerlukan peningkatan berada pada rentang waktu antara 1,8 detik hingga 3,0 detik, yang berarti performa masih cukup baik namun dapat

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

ditingkatkan untuk menjadi lebih responsif. Sementara itu, LCP yang tergolong buruk terjadi jika melebihi 3,0 detik, karena pengguna cenderung merasa halaman lambat dimuat dan berpotensi meninggalkan situs sebelum halaman selesai dimuat (Wp-rocket, 2024b).

Cumulative Layout Shift (CLS)

Berdasarkan tabel interpretasi skor CLS, nilai CLS yang baik adalah kurang dari 0,1, yang berarti elemen-elemen visual pada halaman tetap stabil saat dimuat, sehingga memberikan pengalaman pengguna yang nyaman tanpa gangguan pergeseran tata letak. CLS yang memerlukan peningkatan berada pada rentang 0,1 hingga 0,25, dimana beberapa elemen mungkin masih mengalami pergeseran, meskipun tidak terlalu mengganggu. Sedangkan nilai CLS yang tergolong buruk adalah lebih dari 0,25, yang menunjukkan bahwa banyak elemen berpindah tempat saat halaman dimuat, berpotensi menyebabkan kesalahan klik dan mengganggu interaksi pengguna (Wp-rocket, 2024a).

Speed Index (SI)

Berdasarkan Tabel 3, nilai SI yang cepat berada dalam rentang 0 hingga 1,3 detik, yang menunjukkan bahwa konten halaman ditampilkan dengan cepat dan progresif sehingga memberikan pengalaman visual yang memuaskan bagi pengguna. SI yang sedang berada pada rentang 1,3 hingga 2,3 detik, yang berarti konten masih ditampilkan dalam waktu yang cukup baik, namun bisa dioptimalkan lebih lanjut agar tampilan visual muncul lebih cepat. Sementara itu, SI yang tergolong lambat terjadi jika melebihi 2,3 detik, karena keterlambatan dalam menampilkan konten utama halaman dapat membuat pengguna merasa halaman lambat dan tidak efisien (Chrome Dev*Tool*s team, 2019b).

First Contentful Paint (FCP)

Berdasarkan tabel interpretasi skor FCP, nilai FCP yang termasuk kategori cepat berada dalam rentang 0 hingga 1,8 detik, yang menandakan bahwa konten pertama pada halaman muncul dengan cepat sehingga memberikan kesan responsif kepada pengguna. Kategori sedang berada pada rentang 1,8 hingga 3 detik, menunjukkan bahwa konten pertama masih muncul dalam waktu yang dapat diterima, namun perlu diperbaiki agar lebih cepat. Sedangkan nilai FCP yang masuk dalam kategori lambat adalah jika melebihi 3 detik, yang berarti konten pertama muncul terlalu lambat sehingga dapat mengurangi kenyamanan pengguna dan meningkatkan risiko mereka meninggalkan halaman (Chrome Dev*Tool*s team, 2019a).

Total Blocking Time (TBT)

Berdasarkan tabel interpretasi skor TBT, nilai TBT yang termasuk dalam kategori cepat berada dalam rentang 0 hingga 150 milidetik, menunjukkan bahwa waktu blokir total pada *thread* utama sangat singkat sehingga interaksi pengguna dapat berjalan dengan lancar tanpa hambatan

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

berarti. Kategori sedang berada pada rentang 150 hingga 350 milidetik, yang berarti terdapat jeda blokir yang cukup signifikan sehingga pengalaman interaksi masih bisa diperbaiki untuk menjadi lebih responsif. Sedangkan nilai TBT yang masuk kategori lambat adalah apabila melebihi 350 milidetik, yang mengindikasikan bahwa waktu blokir terlalu lama dan dapat menyebabkan keterlambatan responsivitas aplikasi, sehingga mengganggu kenyamanan pengguna (Chrome Dev*Tool*s team, 2019c).

3. Hasil dan Pembahasan

1) Hasil Pengujian

a) Implementasi Aplikasi

Frontend yang diimplementasikan disini digunakan untuk menjembatani interaksi antara pengguna dengan server.

b) Inisialisasi Next.js

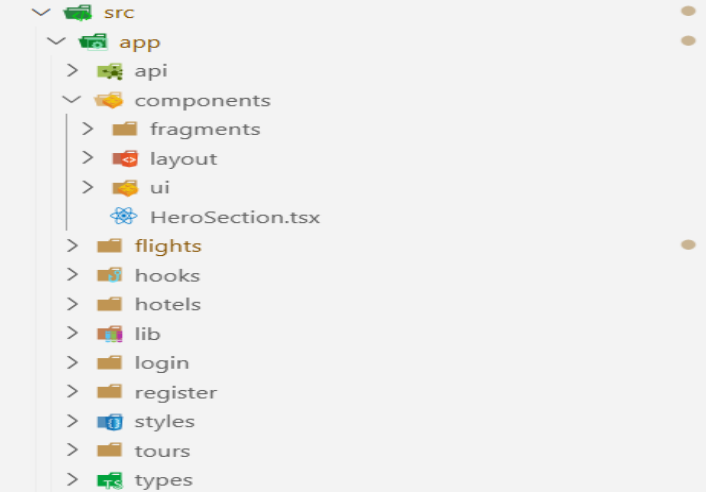
Proses pengembangan dimulai dengan inisialisasi proyek menggunakan perintah berikut.

```
npx create-next-app@latest.
```

Setelah proses ini selesai, Next.js akan secara otomatis membuat struktur dasar proyek beserta konfigurasi awal yang siap untuk dikembangkan lebih lanjut.

c) Struktur Folder

Setelah proses inisialisasi selesai, akan terdapat beberapa folder yang secara otomatis disusun oleh Next.js. Folder-folder ini memiliki nama yang merepresentasikan fungsinya masing-masing, seperti *components* untuk menyimpan komponen antarmuka pengguna (Gambar 5).



Gambar 5. Struktur Folder

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

d) Menyiapkan *Dummy* Data

Sebelum aplikasi terhubung dengan backend sesungguhnya, *Dummy* data perlu disiapkan sebagai data sementara untuk simulasi. Data ini dapat disimpan dalam folder khusus seperti *Dummy*data, dan dituliskan dalam file berformat .ts sebagai objek *TypeScript*. Pendekatan ini memudahkan proses pengujian tampilan dan alur aplikasi tanpa harus bergantung pada data dari server. Contoh dummy data dapat dilihat dalam Gambar 6.

```
import { FlightType } from "app/types/flight.type";

export const flightsDum: FlightType[] = [
  {
    idFlight: 1,
    imageUrl: "https://images.unsplash.com/photo-1569154941061-e231b4725ef1?w=600&auto=format&fit=crop&q=60&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8M
    origin: "Jakarta (CGK)",
    destination: "Bali (DPS)",
    departureDate: "2025-06-10",
    returnDate: "2025-06-15",
    airlineName: "Garuda Indonesia",
    priceStart: 1500000,
  },
```

Gambar 6. *Dummy* Data

e) Menyusun *Page*.tsx

Halaman utama aplikasi disusun dalam file *page*.tsx yang terletak di dalam folder *pages*. Di dalam file ini, struktur *layout*, konten utama, serta pemanggilan data ditulis sesuai dengan alur dan kebutuhan aplikasi.

f) Membuat *Router*

Routing dalam Next.js didasarkan pada struktur file dalam folder *pages*. Untuk navigasi antarhalaman, digunakan komponen *Link* dari next/*Link*, yang memungkinkan transisi halaman tanpa *reload* penuh. *TypeScript* digunakan untuk memastikan bahwa *path* dan parameter *routing* didefinisikan dengan benar, sehingga meminimalkan risiko kesalahan navigasi.

g) Membuat Komponen

Komponen UI dibuat di dalam folder *components*, dengan masing-masing komponen ditulis dalam file .tsx. Setiap komponen menggunakan *Props* yang diketik dengan *interface* atau *type* dari *TypeScript*.

h) Integrasi *Dummy* API

Integrasi dengan *Dummy* API dilakukan dengan cara mengimpor langsung data *dummy* dari file lokal yang berada di folder dataservices/. Pendekatan ini tidak menggunakan fitur API Routes (pages/api) dari Next.js, melainkan mengakses data statis secara langsung di sisi frontend. Data dummy yang sudah diketik menggunakan TypeScript memastikan bahwa struktur data konsisten dan memudahkan proses pengolahan data serta mengurangi potensi terjadinya bug.

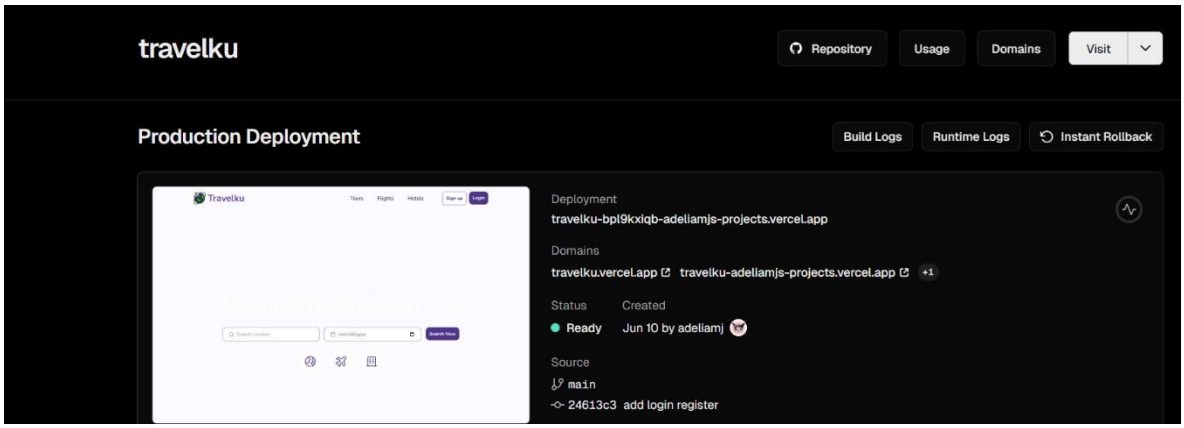
http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

i) Build dan *Hosting* menggunakan Vercel

Sebelum melakukan proses hosting, aplikasi perlu dibangun terlebih dahulu agar siap untuk produksi. Proses *build* ini dilakukan melalui terminal lokal dengan perintah berikut.

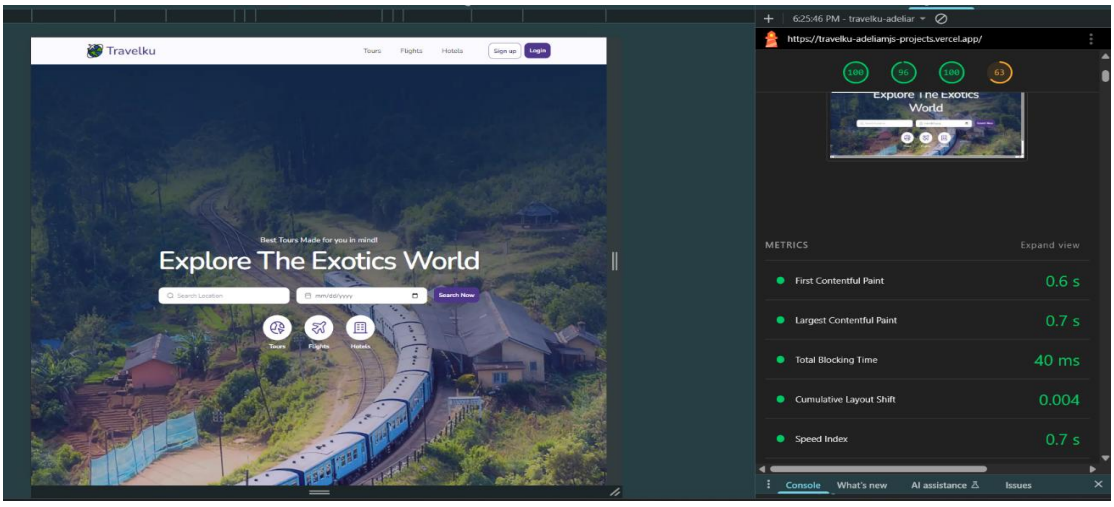
```
npm run build
```

Perintah ini akan mengkompilasi seluruh aset dan dependensi proyek menjadi versi produksi yang optimal, biasanya dalam folder .next (untuk proyek Next.js).



Gambar 7. Hosting Menggunakan Vercel

Setelah proses *build* berhasil, website kemudian dihosting menggunakan layanan Vercel (Gambar 8) Vercel dipilih karena kemudahannya dalam melakukan *deployment* serta kemampuannya dalam menangani website modern seperti Next.js. Next.js sendiri merupakan framework React yang dikembangkan oleh Vercel, sehingga integrasi antara keduanya sangat optimal dan mendukung performa aplikasi secara keseluruhan.



Gambar 8. Proses Pengujian

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

2) Pembahasan Hasil

Pengujian dilakukan terhadap website yang telah diunggah ke platform *hosting* Vercel. Proses ini dijalankan melalui *browser Google Chrome* dalam mode *incognito* guna menghindari pengaruh dari *cache* maupun ekstensi tambahan yang dapat mengganggu akurasi hasil. Pengujian dilakukan menggunakan metode *navigation* sebanyak 8 kali. Hasil dari pengujian diatas disajikan dalam Tabel.4.

Tabel 4. Hasil Pengujian

Parameter Pengujian	LCP	CLS	SI	FCP	TBT
Skor Pengujian 1	0,7s	0,004	0,7s	0,6s	40ms
Skor Pengujian 2	0,4s	0	0,5s	0,3s	20ms
Skor Pengujian 3	0,5s	0	0,4s	0,4s	40ms
Skor Pengujian 4	0,5s	0,004	1,0s	0,4s	50ms
Skor Pengujian 5	0,5s	0	0,5s	0,3s	50ms
Skor Pengujian 6	0,7s	0,004	0,7s	0,6s	40ms
Skor Pengujian 7	0,5s	0	0,5s	0,4s	40ms
Skor Pengujian 8	0,5s	0	0,8s	0,4s	40ms

Berdasarkan Tabel.4, terlihat hasil dari pengujian implementasi *Single Page Application* pada website multi-travel yang di *hosting* pada Vercel menggunakan *Google Lighthouse* dengan mode *navigation*, mode ini menguji website dalam state *default*-nya. Dari delapan hasil pengujian diatas, dapat ditarik nilai *mean* sebagaimana dalam Tabel 5.

1) Analisis Karakteristik Aplikasi

Karakteristik aplikasi SPA ini didapatkan berdasarkan hasil pengujian performa yang telah dilakukan, dengan mengacu pada standar *Core Web Vitals* dari *Google Lighthouse*. Hasil pengujian menunjukkan bahwa aplikasi ini memiliki karakteristik sebagai berikut.

a) Sangat Cepat dalam Menampilkan Konten Utama

Aplikasi ini mampu memuat elemen konten terbesar gambar dan teks dengan sangat cepat, yakni rata-rata hanya dalam waktu 0,5 detik. Ini jauh di bawah ambang batas baik < 1,8 detik. Hal ini mengindikasikan bahwa pengguna akan segera melihat bagian terpenting dari halaman setelah navigasi, serta memberikan kesan aplikasi yang instan dan responsif.

b) Stabilitas Visual yang Luar Biasa

Dengan skor rata-rata CLS hanya 0,0015, yang jauh di bawah ambang batas baik < 0,1 aplikasi ini memiliki stabilitas tata letak visual yang hampir sempurna. Pengguna tidak akan mengalami pergeseran elemen yang mengganggu saat halaman dimuat, memastikan pengalaman yang baik.

http://journal2.um.ac.id/index.php/tekno | ISSN 1693-8739 / 2686-4657

Tabel 5. Rata-rata Hasil Pengujian

Parameter Pengujian	Skor	Keterangan
LCP	0,5s	Skor LCP rata-rata 0,5 detik termasuk dalam kategori hijau (baik) karena jauh di bawah batas 1,8 detik. Ini berarti elemen konten terbesar di halaman web dimuat dengan sangat cepat, memberikan pengalaman pengguna yang sangat optimal dan responsif.
CLS	0,0015	Skor CLS rata-rata 0,0015 termasuk dalam kategori hijau (baik) karena jauh di bawah batas 0,1. Ini menunjukkan bahwa *layout* halaman web sangat stabil saat dimuat, tidak ada pergeseran elemen yang mengganggu, sehingga memberikan pengalaman visual yang nyaman bagi pengguna.
SI	0,6375s	Skor SI rata-rata 0,6375 detik termasuk dalam kategori hijau (cepat) karena berada di bawah batas 1,3 detik. Ini menandakan bahwa konten halaman web ditampilkan secara progresif dan cepat di *viewport*, menciptakan kesan bahwa halaman dimuat dengan efisien dan responsif.
FCP	0,425s	Skor FCP rata-rata 0,425 detik termasuk dalam kategori hijau (cepat) karena jauh di bawah batas 1,8 detik. Hal ini menunjukkan bahwa konten pertama dari halaman web muncul dengan sangat cepat, memberikan kesan responsif instan kepada pengguna dan memastikan mereka tidak menunggu lama untuk melihat konten awal.
TBT	40ms	Skor TBT rata-rata 40 milidetik termasuk dalam kategori hijau (cepat) karena jauh di bawah batas 150 milidetik. Ini mengindikasikan bahwa waktu blokir pada *thread* utama halaman web sangat singkat, memungkinkan interaksi pengguna berjalan dengan sangat lancar dan responsif tanpa hambatan berarti. Pengguna dapat berinteraksi dengan halaman hampir seketika.

c) Tampilan Konten Progresif yang Cepat
Aplikasi ini menampilkan konten di *viewport* secara progresif dengan sangat cepat, rata-rata 0,6375 detik. Skor ini berada dalam kategori cepat < 1,3 detik, menunjukkan bahwa pengguna akan melihat konten halaman dengan cepat dan bertahap, memberikan persepsi t.

d) Responsivitas Awal yang Cepat
Konten pertama dari DOM muncul dengan sangat cepat, rata-rata dalam 0,425 detik. Ini jauh di bawah ambang batas cepat < 1,8 detik. Karakteristik ini memastikan bahwa pengguna segera melihat respons visual dari aplikasi, mengurangi *blank screen* dan meningkatkan persepsi kecepatan *loading*.

e) Interaktivitas Optimal dengan Waktu Blokir Minimal
Dengan rata-rata TBT hanya 40 milidetik, yang jauh di bawah ambang batas cepat < 150 ms, aplikasi ini sangat responsif terhadap *input* pengguna. Ini berarti *thread* utama tidak terblokir dalam waktu lama, memungkinkan pengguna untuk berinteraksi dengan aplikasi hampir seketika tanpa jeda atau *lag*.

Secara keseluruhan, berdasarkan data performa, aplikasi SPA yang diteliti ini memiliki karakteristik sangat cepat, stabil secara visual, dan sangat responsif. Ini menunjukkan bahwa

aplikasi berhasil memberikan pengalaman pengguna yang optimal dari segi kecepatan *loading* dan interaktivitas.

4. Kesimpulan

Berdasarkan hasil penelitian, pengujian, dan pembahasan yang telah dilakukan peneliti, maka dapat ditarik suatu kesimpulan sebagai berikut: (1)

1. Dalam penelitian ini, hasil yang dapatkan menunjukkan bahwa SPA yang diimplementasikan pada sistem pemesanan multi-layanan travel sangat didukung oleh penggunaan Next.js dan *TypeScript*. Dengan arsitektur SPA, seluruh komponen yang dibutuhkan aplikasi akan dimuat di awal, memungkinkan aplikasi untuk secara dinamis menampilkan konten yang berbeda dan berinteraksi dengan pengguna tanpa perlu memuat ulang halaman secara keseluruhan. Next.js, sebagai *framework* React, mendukung hal ini dengan fitur seperti CSR dan kemampuan *pre-rendering* yang meningkatkan kecepatan navigasi antar tampilan. Penggunaan *TypeScript* juga memastikan kode yang lebih *robust* dan mudah dikelola, meminimalkan *bug* dan mempermudah pengembangan fitur-fitur kompleks.
2. Pada tahap pengujian, aplikasi SPA sistem multi-layanan travel dapat berjalan dengan optimal saat di-*host*. Kemudian dilakukan pengujian aplikasi menggunakan *Google Lighthouse*, berfokus pada metrik *Core Web Vitals* untuk mengevaluasi performa.
3. Hasil pengujian menunjukkan bahwa performa dari aplikasi SPA sistem pemesanan multi-layanan travel ini memiliki metrik dengan standar yang dikategorikan sangat cepat sesuai standar *Core Web Vitals*. Setiap metrik pengujian memperoleh skor yang berada pada kategori *"Good"* atau area berwarna hijau, mengindikasikan pengalaman pengguna yang superior dari segi kecepatan *loading*, stabilitas visual, dan responsivitas.

Daftar Rujukan

Amplitude Labs (2022) *The 2022 App vs. Website Trend Report*. https://amplitude.com/guides/2022-app-vs-website-report (Diakses: 9 Juni 2025).

Anugerah Widi, Eko Sediyono, H. (2024) Analisa Performa Website Organisasi Akuatik Menggunakan Automated Software Testing Gtmetrix. 5(August 2023), hal. 25–33. https://doi.org/10.35957/jtsi.v5i2.5925.

Aripin, S. dan Somantri, S. (2021) Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio Mahasiswa, *Jurnal Eksplora Informatika*, 10(2), hal. 148–158. Tersedia pada: https://doi.org/10.30864/eksplora.v10i2.486.

Chrome DevTools team (2019a) *First Contentful Paint*. Tersedia pada: https://developer.chrome.com/docs/lighthouse/performance/first-contentful-paint?hl=id (Diakses: 10 Juni 2025).

Chrome DevTools team (2019b) *Indeks Kecepatan*. Tersedia pada: https://developer.chrome.com/docs/lighthouse/performance/speed-index?hl=id (Diakses: 10 Juni 2025).

Chrome DevTools team (2019c) *Total Waktu Pemblokiran*. Tersedia pada:

https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time?hl=id (Diakses: 10 Juni 2025).

Djauhari, T., Sany, E. dan Sailani (2023) "Penerapan Ui/Ux Pada Website Penjualan Online Toko Bangunan Zulfan Rezqullah.," *Jurnal Akademika*, 16(1), hal. 88–94. Tersedia pada: https://doi.org/10.53564/akademika.v16i1.1121.

Faradilla Ayunindya (2025) *Core Web Vitals: Memahami Metrik Penting untuk Optimasi Website*. Tersedia pada: https://www.hostinger.com/id/tutorial/core-web-vitals#Interaction_to_Next_Paint_INP_–_Kemampuan_Respons_Halaman (Diakses: 10 Juni 2025).

Fauziah, D., Pradana, F. dan Arwan, A. (2019) "Pengembangan aplikasi pemesanan tiket travel berbasis web dengan optimasi jalur penjemputan penumpang (Studi Kasus: Beruang Travel)," *Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(5), hal. 4549–4557.

Kacper Rafalski (2024) *TypeScript vs. JavaScript: Which One Is Best for Your Needs?* Tersedia pada: https://www.netguru.com/blog/typescript-vs-javascript (Diakses: 9 Juni 2025).

Kherina Surya Ningsih, Nur Jamilah Aruan, A.T.A.A.S. (2022) Aplikasi Buku Tamu Menggunakan Fitur Kamera dan AJAX Berbasis Websitwe pada Kantor , *SITek: Jurnal Sains, Informatika, dan Tekonologi*, 1. Tersedia pada: https://doi.org/10.21111/fij.v8i1.8836.

Muttaqin, A.R., Wibawa, A. dan Nabila, K. (2021) Inovasi Digital untuk Masyarakat yang Lebih Cerdas 5.0: Analisis Tren Teknologi Informasi dan Prospek Masa Depan, *Jurnal Inovasi Teknologi dan Edukasi Teknik*, 1(12), hal. 880–886. Tersedia pada: https://doi.org/10.17977/um068v1i122021p880-886.

Nariswari, S.P. dan Kadyanan, I.G.A.G.A. (2023) Analisis Penerapan Single Page Application (SPA) dalam Meningkatkan User Experience pada Sebuah Website, *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, 12(2), hal. 400.

Next.js (2024) *Routing: Pages and Layouts*. Tersedia pada: https://nextjs.org/docs/app/building-your-application/routing/pages-and-layouts (Diakses: 9 Juni 2025).

Nextjs (2024) *Client-side Rendering (CSR)*. Tersedia pada: https://nextjs.org/docs/pages/building-your-application/rendering/client-side-rendering (Diakses: 9 Juni 2025).

Travel Perk (2024) *70+ Online Travel Booking Statistics & Trends*, *Travel Perk*. Tersedia pada: https://www.travelperk.com/blog/online-travel-booking-statistics/ (Diakses: 9 Juni 2025).

Wp-rocket (2024a) *How To Improve Cumulative Layout Shift (CLS) on WordPress*. Tersedia pada: https://wp-rocket.me/google-core-web-vitals-wordpress/improve-cumulative-layout-shift/ (Diakses: 10 Juni 2025).

Wp-rocket (2024b) *How To Improve Largest Contentful Paint (LCP) on WordPress*. Tersedia pada: https://wp-rocket.me/google-core-web-vitals-wordpress/improve-largest-contentful-paint/ (Diakses: 10 Juni 2025).

Zalukhu, A., Swingly, P. dan Darma, D. (2023) "Perangkat Lunak Aplikasi Pembelajaran Flowchart," *Jurnal Teknologi, Informasi dan Industri*, 4(1), hal. 61–70. Tersedia pada: https://ejurnal.istp.ac.id/index.php/jtii/article/view/351.