PENINGKATAN PERFORMA APLIKASI WEB SIAKAD MENGGUNAKAN NEXT.JS DI PUSKORDAT STIMATA

Dinny Wahyu Widarti [1)] **, Mahendra Khamal Akbar** [2)]

[1)] *Teknologi Informasi STMIK PPKIA Pradnya Paramita Malang*
[2)] *Sistem Informasi STMIK PPKIA Pradnya Paramita Malang*
email : dinnywidarti@stimata.ac.id[1)], mahendra_21510020@stimata.ac.id [2)]

Abstraksi

Pengembangan teknologi web yang cepat menuntut adaptasi sistem informasi untuk menjaga keamanan dan performa. Aplikasi SIAKAD berbasis web yang digunakan oleh Puskordat STMIK PPKIA PRADNYA PARAMITA sebelumnya menggunakan AngularJS, yang telah tidak lagi mendapat dukungan resmi, sehingga menimbulkan risiko keamanan dan keterbatasan performa. Penelitian ini bertujuan untuk menerapkan Next.js sebagai solusi modern dalam pengembangan SIAKAD berbasis web. Metode pengembangan yang digunakan adalah Agile dengan pendekatan iteratif dan kolaboratif. Iteratif, karena dengan Agile pengembangan dapat dilakukan dalam siklus pendek. Agile disebut kolaboratif karena sangat mengedepankan komunikasi langsung dengan kerja tim yang erat. Hasil penelitian menunjukkan bahwa penggunaan Next.js meningkatkan keamanan dengan menyembunyikan endpoint API dari sisi klien dan mempercepat waktu muat halaman melalui fitur server-side rendering dan optimasi bawaan lainnya. Dengan demikian, penerapan Next.js pada website Puskordat terbukti efektif dalam meningkatkan keamanan dan performa sistem secara keseluruhan.

Kata Kunci :
Next.js, SSR, Keamanan, *Middleware*.

Abstract

The rapid development of web technology demands the adaptation of information systems to maintain security and performance. The web-based SIAKAD application used by Puskordat STMIK PPKIA PRADNYA PARAMITA previously used AngularJS, which no longer has official support, thus creating security risks and performance limitations. This study aims to implement Next.js as a modern solution in the development of web-based SIAKAD. The development method used is Agile with an iterative and collaborative approach. Iterative, because with Agile development can be done in short cycles. Agile is called collaborative because it prioritizes direct communication with close teamwork. The results of the study show that the use of Next.js improves security by hiding API endpoints from the client side and speeds up page load times through server-side rendering features and other built-in optimizations. Thus, the implementation of Next.js on the Puskordat website has proven effective in improving overall system security and performance.

***Keywords* :**
Next.js, SSR, Security, Middleware

Pendahuluan

STMIK PPKIA Pradnya Paramita yang disingkat dengan STIMATA merupakan salah satu perguruan tinggi swasta di kota Malang. STIMATA memiliki salah satu Unit Pelaksana Teknis (UPT) yaitu Pusat Koordinasi Data (Puskordat), yang menggunakan Sistem Informasi Akademik (SIAKAD) untuk mengelola basis data, perencanaan studi mahasiswa, mencetak hasil studi mahasiswa, serta fungsi-fungsi lainnya yang relevan.

SIAKAD yang sedang berjalan berbasis web dibangun menggunakan AngularJS, yaitu sebuah framework javascript yang cukup popular dan banyak digunakan oleh pengembang di seluruh dunia [1]. Namun saat ini AngularJS sudah tidak lagi dikembangkan dan sudah tidak mendapatkan pembaruan secara resmi, AngularJS support has officially ended as of January 2022 [2]. Hal ini menjadikan framework tersebut kurang ideal untuk digunakan dalam pengambangan aplikasi modern, terutama dari segi keamanan dan efisiensi.

Salah satu kelemahan dari web SIAKAD lama adalah masih terlihat struktur dan *endpoint* API melalui fitur *inspect* pada browser. Hal ini membuka potensi resiko keamanan karena pengguna dapat mengetahui jalur komunikasi data antara *frontend* dengan *backend*. Kondisi inilah yang menjadi focus utama dalam penelitian ini.

Kelemahan lainnya juga teridentifikasi pada beberapa aspek yang diukur menggunakan Google Lighthouse, yaitu aplikasi yang mengukur performa website seperti kecepatan pemanggilan halaman (aspek *performance*), *search engine optimization*

(aspek SEO), struktur program (*coding*) yang diukur dalam aspek *best practice* [3].

Berdasarkan permasalahan dan kelemahan yang diuraikan diatas maka perlu segera dikembangkan ulang agar SIAKAD dapat digunakan dengan baik oleh kampus STIMATA, dalam hal ini admin PUSKORDAT yang akan menggunakan.

Sebagai solusi atas kondisi-kondisi yang dimaksud diatas, pengembangan antarmuka SIAKAD web dilakukan menggunakan Next.js, yaitu framework React modern yang mendukung render sisi *server* (*server-side rendering*) dan memiliki fitur API routing internal. Next.js adalah kerangka kerja, atau alur kerja, yang berasal dari Reactjs yang digunakan untuk membuat aplikasi web sisi klien yang sekarang diterapkan di beberapa situs web di seluruh dunia [4]. Dengan menggunakan Next.js, komunikasi dengan API dapat dilakukan melalui *backend* internal tanpa harus memperlihatkan *endpoint* kepada pengguna. Pendekatan ini memberikan tingkat keamanan yang lebih baik dari sisi frontend. SSR di Next.js memungkinkan *render* halaman dan data di *server* sebelum dikirim ke klien [5].

Next.js juga menawarkan performa yang lebih tinggi melalui optimasi bawaan seperti *pre-rendering* dan pembagian kode otomatis. Dengan demikian, diharapkan versi baru dari website Puskordat mampu memberikan pengalaman pengguna yang lebih cepat, aman, dan efisien. Penggunaan Next.js memberikan keunggulan dalam hal performa dan pengalaman pengguna [6], yang berdampak pada peningkatan skor pada aspek Performance, Best Practice, dan SEO yang diukur menggunakan Google Lighthouse.

Tinjauan Pustaka

Sebagai perbandingan pada penelitian sebelumnya, yaitu penelitian performa web menggunakan Progressive Web Apps (WPA) dengan cara memnggabungkan aplikasi web dengan aplikasi seluler [7].

Penelitian yang juga membangun SIAKAD berbasis web menggunakan metode Waterfall di SMK Plus Nusa Putra menghasilkan SIAKAD yang dapat membantu dalam hal pengolahan data [8]. Kemudian penelitian yang membuat perancangan SIAKAD berbasis web di SMK Negeri 1 Sijunjung memperoleh hasil pengujian oleh pengguna pada hasil kepuasan sebesar 87,78%, serta pada hasil kemanfaatan mendapat 90,63% [9].

Penelitian lain yang menerapan penggunaan Next.js adalah pada penelitian yang berjudul Penerapan Version control system Berbasis Web Menggunakan Next.JS, Nest.JS, Node.JS, dan MongoDB Pada Proses Pengerjaan Skripsi Mahasiswa, menghasilkan system yang telah teruji dengan *black box* [10]. Selain itu juga pada penelitian yang berjudul Rancang Bangun Sistem Pembelian E-Ticket Berbasis Website dengan konsep Server-Side Rendering menggunakan Framework Next Js pada Wisata Telaga Kusuma Jumantono, dengan hasil kelayakan sebesar 89,61% [11].

Metode Penelitian

Pada penelitian ini merupakan penelitian Rekayasa Perangkat Lunak untuk mengembangkan aplikasi SIAKAD berbasis web yang akan digunakan admin PUSKORDAT di Kampus STIMATA. Dalam pengembangan sistem ini, metode Agile dipilih karena menawarkan pendekatan iterative dan fleksibel yang sangat sesuai dengan kebutuhan proyek modern, khususnya untuk pengembangan aplikasi SIAKAD berbasis web yang digunakan oleh admin di UPT Puskordat. Dengan menggunakan Agile dapat memungkinkan tim pengembang untuk merespon perubahan kebutuhan secara cepat dan berkelanjutan, serta memastikan bahwa setiap bagian dari system diuji dan disempurnakan secara bertahap.

Keunggulan utama metode Agile terletak pada kemmapuan memfasilitasi komunikasi intensif, kolaborasi tim yang tinggi, serta penyampaian produk secara incremental dengan kualitas yang terjaga. Dengan metode Agile ini, pengembangan dapat dilakukan dalam beberapa siklus singkat (literasi) yang masing-masing mencakup tahapan perencanaan, implemantasi, pengujian, hingga evaluasi. Hal ini memberikan fleksibilitas lebih dalam pengambilan kepputusan dan penyesuaian terhadap kebutuhan pengguna yang memungkinkan berubah seiring waktu. Agile memiliki keunggulan dalam fleksibilitas dan efisiensi [12].

Langkah-langkah pengembangan aplikasi menggunakan metode Agile seperti yang terlihat pada gambar 1.

Gambar 1. Metode Agile [13]

Pada gambar 1 merupakan siklus iterative dari metode Agile yang terdiri dari beberapa tahapan utama yaitu *Plan*, *Design*, *Develop*, *Test*, *Review*, dan *Lounch*. Setiap tahapan saling berurutan membentuk satu siklus yang dapat diulang untuk menyempurnakan produk secara bertahap. Pendekatan ini memungkinkan pengembangan sistem yang bertahap dan responsif terhadap perubahan kebutuhan, serta memastikan kualitas produk tetap terjaga melalui proses evaluasai dan penyempurnaan yang berkelanjutan. Dalam

implementasinya pada pengembangan SIAKAD berbasis web yang dikelola Puskordat ini, setiap tahap Agile diikuti secara sistematis untuk memastikan hasil yang optimal.

1. Perencanaan (*Planning*)

Tahap perencanaan diawali dengan analisis kebutuhan sistem berdasarkan kelemahan dari versi sebelumnya. Hal ini menjadi alasan mengapa menggunakan Next.js sebagai framework utama. Next.js dipilih karena mendukung *server-side rendering* (SSR) dan internal API routing, yang memungkinkan pemrosesan data dan pembuatan halaman dilakukan di server sebelum dikirim ke pengguna. Dengan cara ini *endpoint* backend tidak langsung terekspos di sisi pengguna, sehingga memberikan perlindungan tambahan dari potensi serangan dan meningkatkan performa tampilan halaman. SSR di Next.js memungkinkan render halaman dan data di *server* sebelum dikirim ke klien [5].

Untuk menunjang efisiensi pengembangan antarmuka, digunakan Tailwind CSS, sebuah framework CSS yang menyediakan kelas-kelas siap pakai untuk berbagai gaya *visual*, seperti warna, ukuran, margin, padding, dan lainnya. Dengan pendekatan ini, pengembang dapat langsung menerapkan gaya *visual* pada elemen HTML tanpa harus menulis kode CSS terpisah, sehingga proses styling menjadi lebih cepat dan konsisten. Tailwind CSS digunakan untuk menciptakan desain antarmuka yang modern dan efisien, memudahkan pengembang [14].

Selain itu dalam perencanaan sistem juga disertakan penerapan *Middleware* sebagai lapisan pengaman tambahan di sisi *server*. *Middleware* berfungsi untuk memproses permintaan pengguna sebelum mencapai handler utama, seperti melakukan pengecekan autentikasi, validasi token, serta pembatasan akses berdasarkan peran pengguna. Dengan demikian, kontrol akses dapat diterapkan lebih terstruktur sejak awal proses permintaan, tanpa membebani logika utama aplikasi. *Middleware* ini ditempatkan di antara permintaan yang masuk dan berfungsi untuk memeriksa izin atau hak akses pengguna sebelum memungkinkan akses ke sumber daya at au fitur tertentu [15].

Cookies dipilih sebagai tempat penyimpanan sementara di browser pengguna. Cookies merupakan salah satu mekanisme penyimpanan di sisi klien yang dimanfaatkan untuk menyimpan refresh token. Browser menyimpan cookies yang diterima dari server di dalam penyimpanan lokal komputer atau perangkat pengguna [16]. Token ini berfungsi sebagai parameter dalam middleware untuk keperluan autentikasi lanjutan. Refresh token dihasilkan oleh backend setelah proses login berhasil dan memungkinkan pengguna memperoleh access token baru tanpa harus melakukan autentikasi ulang. Token tersebut berupa string acak yang telah dikodekan menggunakan format Base64 untuk meningkatkan keamanan saat penyimpanan maupun transmisi data.

Google Lighthouse digunakan untuk menguji performa website berdasarkan empat aspek utama, yaitu *Performance*, *Accessibility*, *Best Practices*, dan SEO. Performance menilai seberapa cepat dan lancar halaman dimuat, sedangkan Accessibility melihat kemudahan penggunaan bagi semua kalangan, termasuk penyandang disabilitas. Best Practices memastikan bahwa aplikasi dibangun dengan standar yang aman dan andal, sementara SEO menilai sejauh mana halaman mudah ditemukan oleh mesin pencari. Keempat aspek ini menjadi acuan penting untuk memastikan aplikasi tidak hanya cepat dan aman, tetapi juga ramah pengguna dan mudah diakses secara luas. *Lighthouse* melakukan serangkaian pengujian terhadap halaman website yang menghasilkan sebuah laporan. Halaman pengujian menentukan skor laporan dari performance, accessibility, best practise, SEO [17].

Sementara itu metode *black-box Testing* digunakan untuk menguji *middleware* dari sudut pandang pengguna tanpa memperhatikan struktur internal kode program. Pengujian dilakukan dengan menyusun berbagai skenario penggunaan umum seperti proses login, navigasi antarmuka, manipulasi data, dan pengamanan akses terhadap halaman-halaman yang dilindungi. Metode ini juga digunakan untuk menilai ketahanan sistem terhadap input tak terduga yang mungkin dapat dimanfaatkan sebagai celah keamanan. Pengujian *black-box* dipilih karena metode ini efektif untuk mengidentifikasi kesalahan pada antarmuka pengguna dan perilaku sistem dalam kondisi nyata. Pemilihan metode Black Box dipilih karena pengujian ini memiliki tujuan untuk memastikan bahwa aplikasi yang telah dibangun memiliki kualitas yang sesuai dengan k riteria yang ditetapkan dan fungsionalitasnya telah dapat berjalan dengan baik sebelum dirilis dan digunakan oleh masyarakat [18].

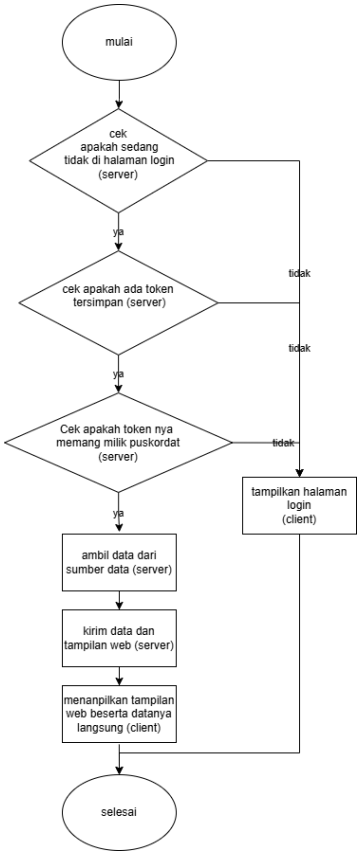
2. Desain (Design)

Tahap desain difokuskan pada perancangan arsitektur sistem yang aman dan efisien. Salah satu aspek utama yang dirancang adalah alur penerapan *server-side rendering* (SSR) dan *Middleware* pada Next.js. SSR memungkinkan halaman dirender di sisi *server* berdasarkan data yang sudah diproses, sehingga pengguna langsung menerima tampilan halaman lengkap tanpa harus menunggu proses rendering di browser mereka. Hal ini tidak hanya meningkatkan kecepatan akses, tetapi juga memberikan keunggulan dari sisi SEO dan kenyamanan penggunaan.

Selain itu, *Middleware* dirancang sebagai lapisan pemeriksaan awal yang bekerja sebelum sistem menampilkan halaman atau memproses data lebih lanjut. Fungsi utama *Middleware* dalam sistem ini adalah untuk melakukan pemeriksaan login, validasi token keamanan, dan pengecekan hak akses pengguna terhadap fitur tertentu. Dengan adanya

proses ini, sistem dapat memastikan hanya pengguna yang berhak yang bisa mengakses halaman atau data tertentu, sehingga keamanan aplikasi menjadi lebih terjaga sejak awal proses.

Untuk memperjelas alur yang telah dirancang, digunakan flowchart seperti pada gambar 2, sebagai media *visual*isasi. Diagram ini menggambarkan tahapan-tahapan yang dilalui oleh setiap permintaan dari pengguna, dimulai dari pemeriksaan melalui *Middleware*, dilanjutkan dengan proses SSR di *server*, hingga akhirnya menghasilkan halaman yang ditampilkan ke pengguna. *Visual*isasi ini membantu tim memahami bagaimana alur data bekerja dan bagian mana yang menjadi titik penting dalam menjaga keamanan serta efisiensi sistem.



Gambar 2. Flowchart penerapan Middleware dan SSR

Dalam pengembangan aplikasi web menggunakan Next.js, kombinasi antara *Middleware* dan *Server-Side* Rendering (SSR) menjadi penting untuk memastikan keamanan akses serta performa tampilan yang optimal. *Middleware* digunakan untuk memfilter pengguna sebelum halaman dirender, sedangkan SSR bertugas merender halaman di sisi *server* agar pengguna mendapatkan tampilan dan data secara instan. Flowchart berikut menjelaskan proses lengkap bagaimana *Middleware* dan SSR bekerja secara berurutan saat pengguna mengakses aplikasi. Next.js bisa melakukan pre-rendering yaitu melalui *Server-Side* Rendering (SSR) [19].

Proses dimulai ketika pengguna mengakses halaman web. Pada tahap ini, sistem mempersiapkan untuk mengevaluasi apakah pengguna memiliki hak akses yang sesuai sebelum menampilkan konten halaman. Kemudian cek apakah sedang tidak di halaman login (*server*). Ambil token dari cookie (*server*). Jika pengguna bukan di halaman login, sistem akan mencoba mengambil token autentikasi yang biasanya disimpan di cookie. Token ini digunakan untuk mengecek apakah pengguna sudah melakukan login sebelumnya. Cek apakah token tersedia (*server*). Setelah token ditemukan, sistem memverifikasi keberadaannya. Jika token tidak tersedia, artinya pengguna belum login dan akan diarahkan kembali ke halaman login. Cek validitas token melalui permintaan verifikasi ke *server* (*server*). Jika token tersedia, langkah selanjutnya adalah memeriksa apakah token tersebut valid. Sistem akan mengirimkan token ke *endpoint* verifikasi untuk memastikan token masih berlaku dan belum kedaluwarsa. Ambil data dari sumber data (*server*). Setelah autentikasi berhasil dan pengguna lolos *Middleware*, *server* akan mulai mengambil data dari sumber yang relevan, misalnya dari database atau API, yang dibutuhkan untuk merender halaman. Kirim data dan tampilan web (*server*). Data yang telah didapatkan kemudian diproses bersama tampilan (UI) oleh *server*. Proses ini disebut dengan *Server-Side* Rendering (SSR), di mana hasil akhirnya berupa HTML siap tampil dikirim ke browser pengguna. Menampilkan tampilan web beserta datanya langsung (*client*). Di sisi pengguna (*client*), halaman web langsung tampil lengkap dengan data yang sudah diolah. Ini membuat pengalaman pengguna lebih cepat karena tidak perlu menunggu data diambil ulang melalui browser. Terakhir proses selesai dan pengguna bisa berinteraksi dengan halaman seperti biasa. Semua pengecekan autentikasi dan pengambilan data sudah dilakukan sebelumnya secara efisien di *server*.

Penjelasan alur ini menggambarkan bagaimana *Middleware* dan SSR saling bekerja untuk menghadirkan pengalaman pengguna yang aman, cepat, dan efisien. Dengan memfilter akses pengguna lebih awal dan langsung merender halaman lengkap dari *server*, aplikasi menjadi lebih andal dan terstruktur dengan baik.

3. Pengembangan (Development)

Tahap pengembangan dimulai dengan membangun struktur proyek menggunakan Next.js. Dalam proses ini, setiap halaman aplikasi dibuat dalam direktori khusus agar pengelolaan kode menjadi lebih teratur dan efisien.

Middleware berperan penting dalam pengamanan aplikasi. Sebelum pengguna mengakses halaman tertentu, sistem terlebih dahulu memeriksa status autentikasi pengguna. Jika pengguna belum login, maka mereka akan diarahkan ke halaman login. Proses ini dilakukan di sisi server, sehingga halaman dan data tidak akan ditampilkan kepada pengguna yang tidak berhak, yang meningkatkan aspek keamanan aplikasi.

Untuk menjaga keamanan sesi pengguna, token autentikasi disimpan di dalam cookies dengan pengaturan keamanan maksimal. Cookies dikonfigurasi menggunakan atribut *HttpOnly*, *Secure*, dan *SameSite=Strict* untuk mencegah akses dari JavaScript, memastikan hanya dikirim melalui koneksi HTTPS, serta melindungi dari serangan Cross-Site Request Forgery (CSRF).

Selain itu, untuk meningkatkan performa pada Google Lighthouse sekaligus memperkuat keamanan aplikasi, diterapkan fitur Server-Side Rendering (SSR). Dengan SSR, data yang dibutuhkan oleh halaman akan diambil terlebih dahulu di sisi server, kemudian digabungkan dengan tampilan sebelum dikirim ke browser. Hal ini membuat proses pemuatan halaman menjadi lebih cepat karena data sudah tersedia saat halaman sampai ke pengguna, serta lebih aman karena data diproses langsung di server.

Untuk menunjang pengembangan antarmuka, Tailwind CSS digunakan sebagai alat bantu styling. Pendekatan berbasis kelas siap pakai dari Tailwind CSS mempercepat proses desain tampilan dan menjaga konsistensi antarmuka, sekaligus meminimalkan penulisan kode CSS manual.

4. Pengujian (*Testing*)

Pengujian performa dilakukan menggunakan Google Lighthouse dan hasilnya dibandingkan dengan versi website sebelumnya yang masih menggunakan AngularJS. Pengujian ini bertujuan untuk melihat peningkatan dari sisi kecepatan, responsivitas, aksesibilitas, kepatuhan terhadap standar pengembangan web, serta optimasi mesin pencari. Hasil skor dari empat aspek utama Performance, Accessibility, Best Practices, dan SEO menunjukkan bahwa website yang dibangun dengan Next.js memberikan pengalaman pengguna yang lebih baik secara keseluruhan dibandingkan versi lama.

Selain aspek performa, pengujian juga dilakukan terhadap keamanan sistem dan middleware dengan pendekatan black-box testing. Pengujian ini berfokus pada bagaimana sistem merespons berbagai skenario serangan atau akses tidak sah tanpa melihat struktur internal kode. Beberapa skenario yang diuji meliputi validasi token, pembatasan akses berdasarkan peran pengguna, dan perlindungan endpoint internal. Hasil pengujian menunjukkan bahwa middleware berhasil membatasi akses secara efektif dan hanya memberikan izin kepada pengguna yang telah terautentikasi dan memiliki hak akses yang sesuai.

5. Penerapan (*Deploy*)

Deploy merupakan proses memindahkan aplikasi dari lingkungan pengembangan (*Develop*ment) ke lingkungan produksi (production) agar dapat digunakan secara nyata oleh pengguna. Selama tahap pengembangan, aplikasi biasanya hanya dapat diakses secara lokal oleh pengembang. Namun, agar aplikasi dapat diakses melalui internet oleh pengguna umum, perlu dilakukan proses *deploy*.

Proses deploy penting karena menjadi tahap akhir yang memastikan aplikasi siap digunakan secara luas dengan infrastruktur yang mendukung akses publik. Selain itu, proses ini juga memungkinkan integrasi sistem dengan lingkungan server yang lebih stabil, aman, dan terkelola. Melalui deploy, sistem dapat diuji dalam kondisi nyata dan mulai memberikan manfaat kepada pengguna sesuai dengan tujuan pengembangannya.

Website Puskordat akan ditempatkan pada sebuah Virtual Private *Server* (VPS). Penggunaan VPS dipilih karena memberikan fleksibilitas tinggi dalam pengelolaan *server*, performa yang lebih baik dibandingkan layanan shared hosting, serta kontrol penuh terhadap konfigurasi sistem. Penempatan pada VPS juga memungkinkan optimalisasi keamanan dan keandalan akses, sehingga website dapat melayani pengguna secara maksimal di lingkungan produksi. Menurut J.Balen dalam Brian ardianto bahwa VPS *Cloud* merupakan teknologi *cloud computing* pada *server* virtual yang menyediakan *server* multifungsi dan fleksibel dengan biaya rendah [20].

6. Tinjauan (*Review*)

Tahap *review* dilakukan setelah proses pengembangan dan pengujian untuk mengevaluasi kembali seluruh komponen aplikasi sebelum dipublikasikan. Tinjauan ini mencakup aspek fungsionalitas, performa, keamanan, serta kesesuaian dengan kebutuhan pengguna yang telah ditentukan pada tahap perencanaan.

Tinjauan dilakukan secara internal oleh tim pengembang untuk memastikan bahwa aplikasi telah memenuhi standar kualitas yang ditetapkan. Jika ditemukan kekurangan atau ketidaksesuaian, perbaikan akan segera dilakukan sesuai dengan prinsip pengembangan Agile, yang menekankan iterasi cepat dan perbaikan berkelanjutan berdasarkan hasil evaluasi. Metode agile merupakan metode yang tahapan dilakukan berulang membentuk suatu siklus untuk mecapai hasil akhir yang diinginkan [21].

7. Tinjauan (*Review*)

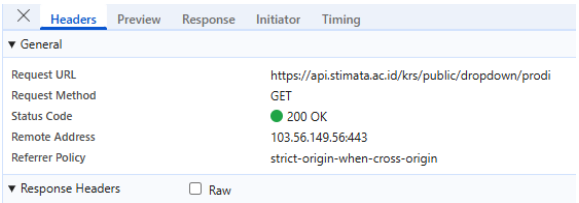
Tahap *launch* merupakan proses peluncuran resmi website ke lingkungan produksi setelah seluruh tahapan pengujian dan tinjauan internal selesai dilaksanakan. Pada tahap ini, website Puskordat yang baru akan menggantikan website lama dan mulai digunakan secara aktif oleh Puskordat STIMATA.

Penggantian website dijadwalkan pada awal tahun akademik 2025/2026 sebagai bagian dari upaya peningkatan layanan informasi dan administrasi. Website baru ini dirancang untuk memberikan peningkatan dari segi performa, keamanan, dan kemudahan penggunaan dibandingkan versi sebelumnya.

Hasil dan Pembahasan

Hasil pertama yang dibahas dalam penelitian ini adalah terkait aspek keamanan endpoint pada

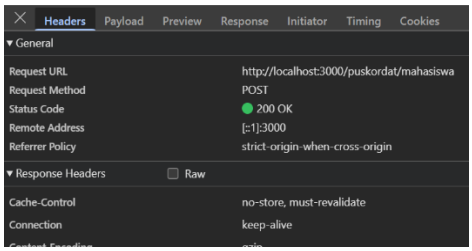
website lama. Keamanan endpoint menjadi perhatian utama karena seluruh proses pengambilan data dilakukan di sisi klien (client-side), sehingga informasi sensitif seperti URL endpoint, parameter permintaan, dan respons API sangat mudah diakses oleh pengguna melalui Developer Tools pada browser. Kondisi ini menimbulkan potensi risiko kebocoran data dan penyalahgunaan akses terhadap API yang tersedia. Oleh karena itu, penting untuk mengevaluasi bagaimana pendekatan lama ini memengaruhi keamanan sistem secara keseluruhan sebelum dilakukan migrasi atau pengembangan ulang dengan teknologi yang lebih aman.



Gambar 3. Proses pengambilan data di SIAKAD lama

Pada Gambar. 3, *endpoint* masih terlihat di website lama karena seluruh proses pengambilan data dilakukan di sisi klien (*client-side*). Ketika data diambil langsung oleh browser pengguna, semua permintaan API dapat diamati melalui *Develop*er tools, termasuk URL *endpoint*, parameter, dan bahkan respons data yang dikembalikan.

Kondisi ini membuat struktur *endpoint* terbuka dan rentan terhadap penyalahgunaan oleh pihak yang tidak berwenang. Tanpa adanya pembatasan atau pengamanan tambahan, informasi penting dapat dengan mudah diakses atau dimanipulasi, sehingga meningkatkan risiko kebocoran data.



Gambar 4. Proses pengambilan data web baru

Pada Gambar. 4, SSR (*Server-Side* Rendering) sudah diterapkan, sehingga proses pengambilan data dilakukan terlebih dahulu di sisi *server* sebelum halaman ditampilkan kepada pengguna. Dengan penerapan ini, link *endpoint* diubah menjadi jalur internal yang hanya dapat diakses dari dalam *server*, bukan langsung dari klien.

Dengan cara ini, struktur *endpoint* API tidak lagi terlihat secara langsung oleh pengguna. Hal ini membantu mengurangi risiko kebocoran data karena akses ke sumber data menjadi lebih terbatas dan terkontrol. Data diambil dan diolah terlebih dahulu di *server*, kemudian hasil akhirnya ditampilkan kepada pengguna dalam bentuk halaman yang sudah lengkap, tanpa mengekspos detail teknis permintaan data ke browser.

Pendekatan ini tidak hanya meningkatkan keamanan, tetapi juga mendukung efisiensi dan kecepatan tampilan melalui SSR. Dibandingkan dengan versi sebelumnya yang menggunakan *client-side* fetching, penggunaan SSR memberikan keunggulan dalam pengamanan struktur aplikasi serta menjaga performa aplikasi tetap optimal.

Selain itu, untuk menambah keamanan, *Middleware* telah diterapkan sebagai lapisan awal pemeriksaan sebelum pengguna dapat mengakses halaman maupun memproses data. *Middleware* ini bertugas untuk memverifikasi keberadaan dan validitas token autentikasi setiap kali ada permintaan masuk, baik itu permintaan akses halaman maupun pemrosesan data melalui API.

Tabel berikut menunjukkan hasil pengujian sistem menggunakan metode black-box *Test*ing terhadap *Middleware* yang telah diterapkan. Pengujian dilakukan dengan berbagai skenario, baik dalam kondisi pengguna telah memiliki token autentikasi maupun tidak. Hasilnya menunjukkan bahwa *Middleware* berhasil menjalankan fungsinya dengan tepat, yaitu mencegah akses tidak sah dan hanya memperbolehkan permintaan dari pengguna yang sudah terautentikasi.

Tabel 1. Hasil pengujian Middleware blackbox

No	Skenario	Token	Di Harapkan	Hasil Aktual	Status
1	Akses halaman login	Tidak	Bisa diakses	Bisa diakses	Berhasil
2	Akses halaman puskordat	Tidak	Dialihkan ke login	Dialihkan ke login	Berhasil
3	Akses halaman puskordat	Ya	Bisa diakses	Bisa diakses	Berhasil
4	Memanggil data	Tidak	Gagal	Gagal	Berhasil
5	Memanggil data	Ya	Data diterima	Data diterima	Berhasil
6	Memasukkan data	Tidak	Gagal	Gagal	Berhasil
7	Memasukkan data	Ya	Data masuk	Data masuk	Berhasil

Berdasarkan Tabel 1, pengujian *Middleware* menunjukkan bahwa sistem berhasil mengatur kontrol akses sesuai skenario yang dirancang. Pada skenario pertama, saat pengguna mengakses halaman login tanpa token, sistem mengizinkan akses sebagaimana mestinya, karena halaman login memang bersifat terbuka untuk semua pengguna, baik yang sudah maupun belum login.

Pada skenario kedua dan ketiga, terlihat bahwa *Middleware* secara efektif membatasi akses ke halaman tertentu seperti halaman puskordat. Tanpa token, pengguna langsung dialihkan ke halaman

login, sementara dengan token yang valid, akses berhasil dilakukan. Hal ini menunjukkan bahwa mekanisme autentikasi bekerja sesuai harapan, dan halaman sensitif tidak dapat diakses tanpa otorisasi.

Pengujian juga dilakukan terhadap permintaan data. Saat pengguna mencoba memanggil data tanpa token, sistem menolak permintaan dan mengarahkannya kembali ke login, sedangkan dengan token yang valid, data dapat diterima dengan baik. Hal ini menunjukkan bahwa *Middleware* tidak hanya mengamankan halaman, tetapi juga melindungi jalur data dari akses yang tidak sah.

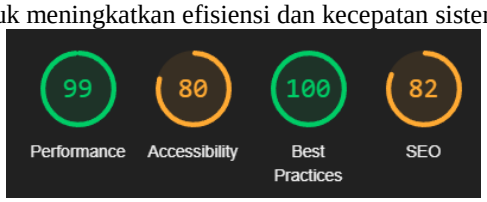
Pada skenario terakhir, yaitu pengujian proses memasukkan data, hasil menunjukkan *Middleware* berhasil mencegah manipulasi data oleh pengguna yang belum terautentikasi. Pengguna tanpa token gagal memasukkan data, sementara pengguna dengan token berhasil melakukan penyimpanan data. Secara keseluruhan, pengujian ini membuktikan bahwa *Middleware* telah diterapkan secara efektif untuk mengamankan akses aplikasi. Ketika digabungkan dengan *Server-Side Rendering* (SSR), proses validasi dan pengambilan data berlangsung lebih aman karena dilakukan terlebih dahulu di sisi *server* sebelum halaman atau data dikirim ke pengguna.

Selain aspek keamanan, performa sistem juga menjadi faktor krusial yang diuji dalam pengembangan aplikasi ini. Pengujian performa dilakukan untuk memastikan bahwa penambahan lapisan keamanan melalui middleware serta penerapan teknologi *Server-Side Rendering* (SSR) tidak memberikan dampak negatif terhadap kecepatan maupun responsivitas aplikasi. Dalam konteks aplikasi berbasis web, performa memiliki peran penting dalam menentukan tingkat kenyamanan pengguna saat berinteraksi dengan sistem.

Gambar 5. Performa SIAKAD web lama

Merujuk pada hasil pengujian menggunakan Google Lighthouse yang ditampilkan pada Gambar 5, diketahui bahwa website versi lama hanya memperoleh skor performa sebesar 37. Skor ini tergolong sangat rendah dan menunjukkan bahwa halaman memerlukan waktu cukup lama untuk dimuat sepenuhnya, terutama pada saat akses pertama. Masalah ini umumnya disebabkan oleh arsitektur yang sepenuhnya mengandalkan pemrosesan di sisi klien (client-side rendering). Dalam pendekatan ini, seluruh data dan tampilan harus dimuat melalui browser pengguna tanpa proses awal dari server. Akibatnya, waktu muat menjadi lebih panjang, khususnya jika koneksi internet pengguna lambat atau perangkat yang digunakan memiliki keterbatasan performa.

Selain itu, website lama belum menerapkan teknik-teknik dasar optimasi, seperti pengurangan beban file atau pengelolaan tampilan secara efisien, sehingga proses rendering menjadi lebih berat. Kombinasi dari berbagai faktor ini menyebabkan performa sistem rendah dan pengalaman pengguna menjadi kurang optimal. Oleh karena itu, diperlukan pendekatan baru untuk meningkatkan efisiensi dan kecepatan sistem.



Gambar 6. Performa SIAKAD web baru

Sebaliknya, pada versi terbaru, yaitu website baru, performa aplikasi menunjukkan peningkatan yang sangat signifikan. Berdasarkan hasil pengujian yang ditunjukkan pada Gambar 6, skor performa website meningkat drastis hingga mencapai angka 99, yang mendekati skor sempurna. Perbaikan ini dicapai melalui penerapan arsitektur Server-Side Rendering (SSR), yang memungkinkan proses pengambilan dan pengolahan data dilakukan terlebih dahulu di sisi server sebelum dikirimkan ke klien. Dengan pendekatan ini, pengguna menerima tampilan halaman yang sudah dirender dari server, sehingga waktu muat menjadi jauh lebih cepat.

Penerapan middleware juga turut berkontribusi dalam peningkatan performa. Middleware bertindak sebagai lapisan kontrol tambahan yang memproses permintaan sebelum diteruskan ke sistem utama. Dengan adanya middleware, beban kerja dapat dikelola lebih efisien, dan proses validasi serta penanganan data dapat dilakukan lebih cepat dan terstruktur.

Hasil pengujian ini menunjukkan bahwa penggunaan SSR dan middleware secara bersama-sama mampu memberikan perbaikan performa yang signifikan. Tidak hanya mempercepat waktu muat halaman, tetapi juga meningkatkan pengalaman pengguna secara keseluruhan melalui sistem yang lebih responsif dan efisien.

Selain peningkatan skor performa, pengujian Lighthouse juga mencatat perbaikan dalam aspek

lainnya. Dari sisi aksesibilitas, terjadi kenaikan skor dari 78 pada versi lama menjadi 80 pada versi baru. Walaupun peningkatannya tidak terlalu besar, hal ini tetap menunjukkan adanya perhatian terhadap kemudahan akses bagi seluruh kalangan pengguna, termasuk mereka yang memiliki kebutuhan khusus.

Peningkatan paling mencolok terjadi pada aspek best practices. Website lama hanya mendapatkan skor 81, sementara pada website baru berhasil mencapai skor maksimal, yaitu 100. Ini menunjukkan bahwa sistem telah dirancang dengan mengikuti praktik terbaik dalam pengembangan web modern, termasuk keamanan data, struktur kode yang baik, dan penggunaan API yang sesuai standar.

Sementara itu, skor SEO pada website baru menunjukkan peningkatan yang cukup signifikan, naik dari nilai 73 menjadi 82. Peningkatan ini merupakan hasil dari berbagai upaya optimasi yang dilakukan selama proses pengembangan, seperti perbaikan struktur konten, peningkatan kecepatan halaman, dan penerapan teknik SEO on-page yang lebih baik. Dengan skor SEO yang lebih tinggi, website tidak hanya menjadi lebih mudah ditemukan oleh mesin pencari, tetapi juga berpotensi menarik lebih banyak pengunjung serta meningkatkan kredibilitas dan reputasi secara online. Hal ini tentu memberikan dampak positif yang signifikan terhadap performa dan keberhasilan website secara keseluruhan.

Secara keseluruhan, hasil perbandingan antara versi lama dan baru menegaskan bahwa pembaruan sistem memberikan dampak yang sangat positif, tidak hanya dari sisi keamanan dan arsitektur, tetapi juga dalam hal performa dan pengalaman pengguna. Perubahan pendekatan teknis yang dilakukan terbukti mampu meningkatkan efisiensi sistem secara menyeluruh dan memberikan fondasi yang lebih kuat untuk pengembangan aplikasi ke depannya.

Kesimpulan dan Saran

Kesimpulan dari penelitian ini menegaskan bahwa penerapan Server-Side Rendering (SSR) dan middleware pada website baru berhasil memberikan peningkatan yang sangat signifikan pada aspek keamanan dan performa jika dibandingkan dengan website lama yang masih mengandalkan proses pengambilan data di sisi klien. Dengan menggunakan SSR, proses pengambilan dan pengolahan data dilakukan di sisi server terlebih dahulu sebelum halaman dikirim ke pengguna, sehingga endpoint API tidak lagi terekspos langsung ke browser. Hal ini secara efektif mengurangi risiko kebocoran data dan potensi penyalahgunaan akses oleh pihak yang tidak berwenang. Middleware yang diterapkan juga terbukti sangat efektif dalam melakukan kontrol akses dengan cara memverifikasi token autentikasi pada setiap permintaan yang masuk, baik saat mengakses halaman maupun saat memproses data melalui API. Pengujian menggunakan metode black-box menunjukkan bahwa sistem dapat membatasi akses sesuai skenario yang diharapkan, dimana hanya pengguna yang sudah terautentikasi yang diperbolehkan mengakses sumber daya tertentu.

Selain aspek keamanan, performa website juga menunjukkan peningkatan luar biasa setelah implementasi SSR dan middleware. Berdasarkan pengujian Google Lighthouse, skor performa website meningkat drastis dari 37 pada versi lama menjadi 99 pada versi baru, yang menunjukkan waktu muat halaman jauh lebih cepat dan pengalaman pengguna yang lebih responsif. Hal ini dikarenakan SSR memungkinkan halaman dikirimkan dalam bentuk sudah ter-render, sehingga pengguna tidak perlu menunggu proses pemuatan data yang lama di sisi klien. Selain itu, penerapan middleware turut membantu mengelola beban kerja secara lebih efisien dengan memproses permintaan secara terstruktur sebelum diteruskan ke sistem utama. Peningkatan juga terlihat pada aspek aksesibilitas, best practices, dan SEO, yang menandakan bahwa website baru dirancang sesuai dengan standar pengembangan web modern. Skor aksesibilitas naik dari 78 menjadi 80, skor best practices meningkat dari 81 menjadi 100, dan skor SEO meningkat dari 73 menjadi 82. Hal ini menunjukkan perhatian yang lebih besar terhadap kemudahan akses bagi semua pengguna, keamanan data, serta optimasi konten agar lebih mudah ditemukan oleh mesin pencari.

Secara keseluruhan, hasil perbandingan antara versi lama dan baru membuktikan bahwa perubahan arsitektur dan penerapan teknologi yang lebih modern mampu memberikan dampak positif yang menyeluruh. Tidak hanya meningkatkan keamanan sistem dengan cara yang lebih sistematis dan terkontrol, tetapi juga meningkatkan performa sehingga pengalaman pengguna menjadi lebih optimal. Pendekatan baru ini memberikan fondasi yang kuat untuk pengembangan aplikasi lebih lanjut di masa depan, dengan sistem yang lebih efisien, aman, dan sesuai dengan standar teknologi terkini. Dengan demikian, penelitian ini menegaskan pentingnya adopsi teknologi yang tepat dalam pembangunan aplikasi web demi mencapai keseimbangan antara keamanan, performa, dan kenyamanan pengguna.

Berdasarkan hasil penelitian ini menunjukkan bahwa dengan mengembangkan SIAKAD baru menggunakan Next.js dapat meningkatkan performa aplikasi Web SIAKAD untuk PUSKORDAT.

Daftar Pustaka

[1] A. Leo, V. Kuswanto, and L. Damayanti, “Mobile Commerce untuk Sales Order dan Tracking Order berbasis MVC,” *ALGOR*, vol. 4, no. 1, pp. 118–126, doi: https://doi.org/10.31253/algor.v4i1.1748.

[2] AngularJS, “Version Support Status.” The MIT License. [Online]. Available: https://docs.angularjs.org/misc/version-support-status

[3] M. Daffa Al-farel and A. R. Dzikrillah, “Evaluasi Framework Pengembangan Web KinarBhusana:

Studi Performance, SEO, dan Accessibility Menggunakan Laravel dan Bootstrap," *Bulletin of Computer Science Research*, vol. 5, no. 3, pp. 235–242, doi: https://doi.org/10.47065/bulletincsr.v5i3.499.

[4] H. Setyani, J. Rahmadian, R. Syawali, A. P. Ningrum, and S. Ali, "PERANCANGAN APLIKASI WEDDING ORGANIZER BERBASIS WEBSITE PADA RIRA GRIYA RIAS PENGANTIN," *JATI*, vol. 8, no. 6, p. 11340, doi: https://doi.org/10.36040/jati.v8i6.11340.

[5] A. Sasongko, B. Lailiah, P. Surya Hadikusumah, and Y. Marien, "Implementasi Agile Pengembangan Sistem Informasi Karyawan Berbasis Next. Js dan PostgreSQL: Studi Kasus Yayasan Almadani," *Kesatria: Jurnal Penerapan Sistem Informasi (Komputer dan Manajemen)*, vol. 6, no. 1, pp. 292–301.

[6] R. P. A. Nugroho, "Meningkatkan Performa Frontend dengan Menggunakan Framework Next.Js dalam Pengembangan Website," *JOCHAC*, vol. 2, no. 2, pp. 14–19, doi: https://doi.org/10.64163/jochac.v2i2.31.

[7] E. Pratama, "Mengoptimalkan Performa Website dengan Progressive Web Apps (PWA)," *Circle Archive*, vol. 1, no. 6, pp. 1–7.

[8] B. A. Adli and S. Abdullah, "PERANCANGAN SISTEM INFORMASI AKADEMIK (SIAKAD) BERBASIS WEB MENGGUNAKAN METODE WATERFALL STUDI KASUS SMK PLUS NUSA PUTRA," *Prosiding Seminar Nasional Teknologi Informasi, Mekatronika, Dan Ilmu Komputer*, vol. 1, no. 2022, [Online]. Available: http://prosiding.sentimeter.nusaputra.ac.id/index.php/prosiding/article/view/7

[9] M. Z. Arsyad, T. Mary, and S. Junaidi, "Perancangan Sistem Informasi Akademik (SIAKAD) Berbasis Web di SMK Negeri 1 Sijunjung," *Jurnal Ilmiah Sistem Informasi dan Teknik Informatika (JISTI)*, vol. 8, no. 1, pp. 65–75, doi: 10.57093/jisti.v8i1.275.

[10] A. R. Soplanit, A. D. Saputro, R. M. Kmurawak, and M. R. Sampebua, "Penerapan Version control system Berbasis Web Menggunakan Next.JS, Nest.JS, Node.JS, dan MongoDB Pada Proses Pengerjaan Skripsi Mahasiswa," *Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)*, vol. 8, no. 2, pp. 361–370.

[11] A. Nugroho, K. Prihandani, and R. Mayasari, "RANCANG BANGUN SISTEM PEMBELIAN E-TICKET BERBASIS WEBSITE DENGAN KONSEP SERVER-SIDE RENDERING MENGGUNAKAN FRAMEWORK NEXT JS PADA WISATA TELAGA KUSUMA JUMANTONO," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 4, pp. 5771–5777, doi: https://doi.org/10.36040/jati.v8i4.

[12] C. Ramadhan, M. A. Sanubekti, and D. Amalia, "Penerapan Metodologi Agile dalam Pengembangan Perangkat Lunak," *Router : Jurnal Teknik Informatika Dan Terapan*, vol. 3, no. 2, doi: https://doi.org/10.62951/router.v3i2.411.

[13] S. B. Atim, "Permodelan Sistem Informasi Penjualan Barang Berbasis Website Menggunakan Metode Agile," *Journal of Artificial Intelligence and Technology Information (JAITI)*, vol. 2, no. 1, pp. 14–25, doi: https://doi.org/10.58602/jaiti.v2i1.104.

[14] K. A. bayu W. Kesuma, I. N. Y. A. Wijaya, and I. G. J. E. P. Putra, "Implementasi Next.Js, Typescript, Dan Tailwind Css Untuk Pengembangan Aplikasi Frontend Sistem Inventory Perusahaan Apar (Studi Kasus: CV Indoka Surya Jaya)," *Jikom: Jurnal Informatika Dan Komputer*, vol. 14, no. 2, pp. 95–108, doi: https://doi.org/10.55794/jikom.v14i2.195.

[15] R. M. Ulah and I. M. Suartana, "Implementasi Session Management Pada Website Magang Menggunakan Teknologi MERN," *Journal of Informatics and Computer Science (JINACS)*, vol. 6, no. 3, pp. 765–777, doi: https://doi.org/10.26740/jinacs.v6n03.p765-777.

[16] S. D. Hermawan, "COOKIES DAN PRIVASI : SEBERAPA AMAN DATA PENGGUNA DITANGAN WEBSITE." researchgate.net. [Online]. Available: https://www.researchgate.net/profile/Syahidan-Hermawan/publication/390265289_COOKIES_DAN_PRIVASI_SEBERAPA_AMAN_DATA_PENGGUNA_DITANGAN_WEBSITE/links/67e65d4e43ec6369e204de25/COOKIES-DAN-PRIVASI-SEBERAPA-AMAN-DATA-PENGGUNA-DITANGAN-WEBSITE.pdf

[17] M. A. S. Ajay, A. Novianti, and A. Hartaman, "Perancangan Dan Implementasi Website Untuk Monitoring Dan Controlling Dry Cabinet," *eProceedings of Applied Science*, vol. 6, no. 2, pp. 2364–2374.

[18] P. Saman and C. I. Ratnasari, "Pengujian Black Box Pada Aplikasi Pembelajaran Bahasa Mandarin Berbasis Android," *Jurnal Ilmiah Intech : Information Technology Journal of UMUS*, vol. 4, no. 1, pp. 10–22, doi: https://doi.org/10.46772/intech.v4i01.637.

[19] D. A. Anugerah and S. Kosasi, "Penerapan Next.js dan GraphQL dalam Pengembangan Mobile Web Sistem Informasi Manajemen Puskesmas," *CSRID (Computer Science Research and Its Development Journal)*, vol. 16, no. 3, pp. 247–258, doi: https://doi.org/10.22303/csrid-.16.3.2024.247-258.

[20] B. Ardianto, M. T. W. Pangesti, and P. T. Pungkasanti, "Penerapan Metode ROC dan MAIRCA Dalam Pemilihan Web Hosting VPS Cloud," *Teknika*, vol. 13, no. 3, pp. 396–402, doi: https://doi.org/10.34148/teknika.v13i3.943.

[21] Z. Amin and N. Pasha, "Penerapan Metode Design Thinking dan Agile dalam Rancang Bangun Aplikasi Penjualanku," *Journal of Information System Research (JOSH)*, vol. 4, no. 3, pp. 755–766, doi: DOI%2010.47065/josh.v4i3.3117.