Published online on the page: https://journal.upp.ac.id/index.php/rjti/index

R J T I

Riau Jurnal Teknik Informatika

| E - ISSN (*Online*) 2829-2529 |

Pengembangan dan Pengujian Performa *Progressive Web App* pada Sistem Reservasi Ruangan Fakultas Teknik Universitas Sam Ratulangi

Veronica Waeo[1,*], Alwin Melkie Sambul[2], Salaki Reynaldo Joshua[3]

[1,2,3]Universitas Sam Ratulangi, Kota Manado, Indonesia

Informasi Artikel

Sejarah Artikel:
Submit: 14 Juli 2026
Revisi: 07 Juli 2026
Diterima: 21 Juli 2026
Diterbitkan: 23 Juli 2026

Kata Kunci

Progressive Web App, Google Lighthouse, Rapid Application Development, Selenium, Sistem Reservasi Ruangan

Korespondensi

E-mail:

veronicawaeo026@student.unsrat.ac.id*

A B S T R A C T

The manual room reservation process at the Faculty of Engineering, Sam Ratulangi University leads to uncertain application status, scheduling conflicts, and limited transparency. Existing reservation practices also lack offline accessibility and installable web capabilities. This study aims to develop and evaluate a Progressive Web App (PWA)-based room reservation system to improve accessibility while maintaining performance. The system was developed using the Rapid Application Development method and evaluated through automated black box testing with Selenium and performance testing using Google Lighthouse. Performance measurements were conducted on four representative pages using Google Chrome and Lighthouse under online and offline scenarios, with each test repeated three times on the same desktop environment. The PWA achieved average performance scores of 100 for cache-offline and cache-online scenarios, 98.5 for no-cache offline, and 96 for no-cache online. The non-PWA version achieved average scores of 99.75 and 96.25 under online conditions but could not function offline. These results demonstrate that the proposed PWA provides installation capability and limited offline access while maintaining online performance comparable to a conventional web application, making it a practical solution for room reservation services in higher education.

Abstrak

Proses reservasi ruangan secara manual di Fakultas Teknik Universitas Sam Ratulangi menimbulkan ketidakpastian status pengajuan, bentrok jadwal, dan keterbatasan transparansi. Selain itu, sistem yang digunakan belum mendukung akses offline maupun kemampuan instalasi seperti aplikasi. Penelitian ini bertujuan mengembangkan dan mengevaluasi sistem reservasi ruangan berbasis *Progressive Web App* (PWA) untuk meningkatkan aksesibilitas tanpa mengurangi performa. Sistem dikembangkan menggunakan metode *Rapid Application Development* dan dievaluasi melalui *automated black box testing* menggunakan Selenium serta pengujian performa menggunakan Google Lighthouse. Pengujian dilakukan pada empat halaman menggunakan Google Chrome dan Google Lighthouse pada lingkungan desktop dengan skenario online dan offline, serta setiap pengujian diulang sebanyak tiga kali. Hasil pengujian menunjukkan bahwa PWA memperoleh rata-rata skor performa 100 pada skenario *cache offline* dan *cache online*, 98,5 pada *no cache offline*, serta 96 pada *no cache online*. Versi non-PWA memperoleh

rata-rata skor 99,75 dan 96,25 pada kondisi *online*, namun tidak dapat beroperasi secara *offline*. Hasil tersebut menunjukkan bahwa PWA mampu menyediakan kemampuan instalasi dan akses *offline* terbatas tanpa menurunkan performa *online* secara signifikan sehingga layak diterapkan pada layanan reservasi ruangan di lingkungan perguruan tinggi.

1. Pendahuluan

Perkembangan teknologi web mengubah situs dari media penyaji informasi menjadi lingkungan aplikasi yang interaktif, adaptif, dan dapat digunakan lintas perangkat. Evolusi tersebut berpengaruh langsung pada sektor pendidikan karena layanan akademik dan administrasi membutuhkan akses informasi yang cepat, akurat, serta mudah dijangkau [1], [2]. Sistem berbasis web juga semakin banyak digunakan untuk mengelola peminjaman fasilitas kampus karena mampu memusatkan data, mengurangi pekerjaan administratif, dan memperjelas ketersediaan sumber daya [3], [4].

Fakultas Teknik Universitas Sam Ratulangi memiliki 6 gedung aktif dengan beberapa ruangan pada masing-masing gedung tersebut yang digunakan untuk kegiatan akademik dan nonakademik, meliputi auditorium, ruang sidang, ruang kelas, laboratorium, dan lainnnya. Berdasarkan wawancara dengan staf Bidang Umum, proses reservasi masih dilakukan melalui penyerahan surat dan disposisi berjenjang. Dokumen berpindah dari peminjam ke Bidang Umum, pimpinan fakultas, dan pengelola ruangan sebelum keputusan diterima. Alur tersebut menimbulkan waktu tunggu yang tidak pasti, kesulitan melacak status pengajuan, risiko bentrok jadwal, serta ketergantungan pada dokumen fisik.

Hasil wawancara dengan beberapa mahasiswa menunjukkan bahwa proses reservasi secara manual umumnya memerlukan waktu sekitar 1–3 hari kerja karena dokumen harus melalui beberapa tahapan disposisi. Selama proses tersebut, mahasiswa kesulitan mengetahui perkembangan pengajuan karena tidak tersedia media untuk memantau status reservasi secara langsung. Akibatnya, peminjam harus melakukan konfirmasi berulang kepada petugas Bidang Umum untuk memastikan apakah surat masih diproses, telah disetujui, atau ditolak. Selain itu, berdasarkan hasil wawancara ditemukan kasus ketika mahasiswa mengajukan peminjaman salah satu ruangan di Gedung Jurusan Teknik Elektro untuk kegiatan Unit Kegiatan Mahasiswa (UKM), namun pengajuan tersebut ditolak karena pada waktu yang sama ruangan telah dijadwalkan untuk kegiatan perkuliahan. Kondisi tersebut menunjukkan bahwa peminjam tidak memiliki akses terhadap informasi ketersediaan ruangan secara *real-time* sebelum mengajukan permohonan. Oleh karena itu, sistem reservasi terpusat diperlukan agar ketersediaan ruangan dapat diperiksa lebih awal dan status pengajuan dapat dipantau secara transparan.

Permasalahan menjadi lebih kompleks karena ruangan berada pada gedung dan unit pengelola yang berbeda. Reservasi ruang umum memerlukan verifikasi Bidang Umum dan persetujuan pimpinan fakultas, sedangkan laboratorium juga melibatkan kepala laboratorium atau jurusan. Tanpa basis data jadwal yang terintegrasi, pemeriksaan ketersediaan masih dilakukan secara manual melalui komunikasi dan dokumen yang tersebar sehingga memperbesar risiko bentrok jadwal dan keterlambatan informasi. Oleh karena itu, sistem reservasi yang terintegrasi diperlukan untuk mendukung validasi jadwal, alur persetujuan, notifikasi status, serta pengelolaan data gedung dan ruangan.

Salah satu pendekatan yang dapat memperluas kemudahan akses aplikasi web *adalah Progressive Web App* (PWA). PWA menggunakan teknologi web modern untuk memberikan pengalaman yang menyerupai aplikasi terpasang, antara lain melalui *web app manifest*, *service worker*, penyimpanan *cache*, dan koneksi aman HTTPS [5], [6]. PWA dapat dipasang pada layar utama, memuat aset yang telah

disimpan dengan lebih cepat, serta menyediakan akses terbatas saat jaringan tidak tersedia [7]. Karakteristik ini sesuai dengan sistem reservasi ruangan yang perlu dibuka berulang kali oleh pengguna untuk memeriksa jadwal dan status pengajuan.

Penelitian terdahulu menunjukkan bahwa sistem reservasi berbasis web mampu meningkatkan efisiensi administrasi dan mengurangi konflik jadwal [3], [4]. Penelitian lain juga menunjukkan bahwa Progressive Web App mampu meningkatkan aksesibilitas, mendukung penggunaan lintas perangkat, serta memberikan performa yang baik melalui pemanfaatan *service worker* dan *cache* [5]–[10]. Ringkasan penelitian terdahulu disajikan pada Tabel 1.

Tabel 1. Ringkasan Penelitian Terdahulu

No	Nama Peneliti	Judul Penelitian	Hasil Penelitian	Perbedaan dengan Penelitian Ini
1	T. E. Kotambunan and T. Witono	Sistem Manajemen Peminjaman Ruangan untuk Kegiatan Kemahasiswaan Berbasis Website di Universitas Kristen Maranatha	Memusatkan proses pengajuan dan informasi jadwal	Tidak menerapkan PWA dan tidak mengevaluasi performa aplikasi.
2	F. A. Pangestu, F. Athallah, A. Sofwan, and Y. Christyono	Pengembangan Aplikasi Sistem Informasi Reservasi Ruangan Berbasis Website	Mengurangi duplikasi jadwal dan meningkatkan administrasi	Berfokus pada implementasi sistem, tanpa pengujian PWA maupun perbandingan performa.
3	R. A. Saputra, I. A. Kautsar, N. Ariyanti, and C. Taurusta	*Implementation Of Progressive Web Apps On Digital Freelance Platform With Rapid Application Development Method*	Meningkatkan aksesibilitas aplikasi berbasis web	Tidak membandingkan PWA dengan non-PWA dan tidak menguji berbagai skenario *cache* maupun *offline*.
4	G. Matiini, R. Setiyadi, A. Setiawan, and M. Ramli	Pengembangan Aplikasi *Progressive Web Application* (PWA) untuk Pembelajaran dan Evaluasi Kelas *English Grammar Online Course*	Mendukung akses lintas perangkat dan meningkatkan pengalaman pengguna	Objek penelitian bukan sistem reservasi serta tidak mengevaluasi performa menggunakan Google Lighthouse.
5	R. J. P. Joarno, M. Fajar, and A. Yunus	Implementasi *Progressive Web Apps* pada Website GetHelp Menggunakan Next.js	Mengevaluasi performa menggunakan Google Lighthouse	Hanya mengevaluasi implementasi PWA tanpa pembanding non-PWA dan tanpa pengujian fungsional otomatis.

Berdasarkan ringkasan penelitian terdahulu pada Tabel 1, masih terdapat beberapa kesenjangan penelitian. Sebagian besar penelitian berfokus pada implementasi PWA tanpa membandingkannya secara langsung dengan versi non-PWA yang memiliki fungsi, antarmuka, dan data yang sama. Selain itu, pengujian performa umumnya hanya dilakukan pada kondisi *online*, sedangkan kombinasi pengujian fungsional otomatis menggunakan Selenium dan pengujian performa menggunakan Google Lighthouse pada berbagai skenario *cache*, tanpa *cache*, *online*, dan *offline* masih jarang dilakukan. Akibatnya, pengaruh implementasi PWA terhadap performa dan ketersediaan akses belum dapat dievaluasi secara menyeluruh.

Perbandingan yang setara memerlukan dua versi dengan fungsi bisnis, antarmuka, data, perangkat, dan prosedur pengujian yang sama, sedangkan perbedaannya hanya terletak pada komponen PWA. Pendekatan ini membantu memisahkan pengaruh *manifest*, *service worker*, dan *cache* dari pengaruh fitur aplikasi lainnya. Evaluasi juga perlu menjelaskan batas akses *offline* secara konkret karena istilah *offline* sering disalahartikan sebagai kemampuan menyelesaikan seluruh transaksi tanpa jaringan. Pada sistem reservasi, data jadwal terbaru, autentikasi, unggah surat, pengiriman formulir, dan persetujuan tetap bergantung pada *server*. Oleh sebab itu, penelitian ini menilai performa sekaligus ketersediaan antarmuka, mekanisme *fallback*, instalasi, dan pemulihan koneksi.

Berdasarkan latar belakang dan kesenjangan penelitian tersebut, penelitian ini berfokus pada dua rumusan masalah, yaitu pengembangan sistem reservasi ruangan Fakultas Teknik Universitas Sam Ratulangi berbasis *Progressive Web App* yang mampu mendukung proses reservasi secara lebih efektif dan transparan, serta perbandingan performa antara sistem berbasis PWA dan non-PWA pada kondisi *cache*, tanpa *cache*, *online*, dan *offline* berdasarkan hasil pengujian menggunakan Google Lighthouse.

Berdasarkan rumusan masalah yang ada penelitian ini bertujuan mengembangkan prototipe sistem reservasi ruangan Fakultas Teknik Universitas Sam Ratulangi berbasis *Progressive Web App*, memverifikasi kesesuaian fungsi menggunakan *automated black box testing* berbasis Selenium, serta membandingkan performa sistem PWA dan non-PWA menggunakan Google Lighthouse pada berbagai skenario pengujian.

Kontribusi penelitian ini terletak pada penyajian evaluasi yang membandingkan dua versi sistem dengan fungsi bisnis, antarmuka, data, perangkat, dan prosedur pengujian yang sama sehingga perbedaan hasil hanya dipengaruhi oleh implementasi komponen PWA. Selain itu, penelitian ini mengevaluasi 6 skenario pengujian terhadap empat halaman utama, termasuk kemampuan instalasi aplikasi dan akses *offline* terbatas. Hasil penelitian diharapkan menjadi referensi bagi pengembangan layanan administrasi kampus yang membutuhkan aksesibilitas dan performa yang stabil.

2. Metode Penelitian

2.1. Tahapan Penelitian

Tahapan penelitian disusun mengikuti alur perencanaan kebutuhan, desain sistem, pengembangan, implementasi dan pengujian, serta analisis hasil. Pada tahap perencanaan kebutuhan dilakukan studi literatur, observasi, dan wawancara untuk mengidentifikasi proses reservasi yang berjalan dan kebutuhan pengguna. Tahap desain sistem menerjemahkan kebutuhan tersebut ke dalam rancangan proses, basis data, dan antarmuka. Pemodelan menggunakan diagram UML membantu menjelaskan interaksi aktor dan urutan aktivitas sistem [18].

Tahap pengembangan merealisasikan rancangan menjadi aplikasi, dan tahap implementasi dan pengujian memastikan fungsi serta performa sistem. Hasil pengujian kemudian dianalisis dengan membandingkan PWA dan non-PWA pada kondisi yang setara. Urutan penelitian ditunjukkan pada Gambar 1.

Data kebutuhan diperoleh dari dokumentasi proses, hasil wawancara, dan pengamatan terhadap alur peminjaman yang berjalan. Kebutuhan kemudian dikelompokkan menjadi kebutuhan pengguna, administrator, superadministrator, serta kebutuhan nonfungsional. Selama pengembangan, rancangan antarmuka dan proses diperiksa kembali agar sesuai dengan alur persetujuan fakultas. Evaluasi dilakukan setelah fungsi utama stabil. Hasil *black box* dinilai berdasarkan kesesuaian *output* aktual dengan *output* yang diharapkan, sedangkan hasil Lighthouse dihitung dari rata-rata tiga pengulangan pada tiap kombinasi halaman dan skenario. Pemisahan tahap tersebut menjaga agar kegagalan fungsi tidak tercampur dengan variasi pengukuran performa.

Gambar 1. Tahapan Penelitian

2.2. Pengembangan Sistem

Pengembangan sistem dilakukan menggunakan metode *Rapid Application Development* (RAD) karena metode ini mendukung proses iteratif melalui perencanaan kebutuhan, desain yang dapat diuji

dan disempurnakan, pengembangan, serta implementasi [12]. Sistem dibangun menggunakan Next.js, database PostgreSQL, Prisma ORM, dan Tailwind CSS. Sistem reservasi ini menyediakan autentikasi akun UNSRAT, pengecekan ketersediaan ruangan, formulir reservasi, riwayat dan status pengajuan, alur persetujuan, notifikasi, serta pengelolaan gedung dan ruangan.

Komponen PWA diterapkan melalui *web app manifest*, *app shell*, dan *service worker*. Manifest menyimpan identitas aplikasi, ikon, *start URL*, warna tema, dan mode *standalone*. Service worker melakukan *precache* terhadap aset penting dan menerapkan strategi *cache first* untuk aset statis, *network first* untuk data dinamis, serta *network first with offline fallback* untuk navigasi halaman. Pendekatan ini menjaga keterbaruan data saat jaringan tersedia dan menyediakan respons alternatif ketika koneksi terputus [9], [19].

Struktur data sistem dirancang menggunakan *Entity Relationship Diagram* (ERD) untuk menggambarkan hubungan antarentitas yang digunakan dalam sistem reservasi ruangan. ERD membantu memastikan integritas data serta mendukung proses implementasi basis data pada PostgreSQL.

2.3. Lingkungan Pengujian

Sebelum proses pengujian dilakukan, ditetapkan lingkungan pengujian agar seluruh skenario dilaksanakan pada kondisi yang konsisten dan hasil pengukuran dapat direplikasi. Seluruh pengujian dilakukan menggunakan perangkat, sistem operasi, peramban, serta konfigurasi yang sama untuk versi PWA maupun non-PWA. Pengujian performa menggunakan Google Lighthouse dijalankan melalui Chrome DevTools pada mode *desktop*, sedangkan pengujian fungsional menggunakan Selenium WebDriver. Konfigurasi lingkungan pengujian disajikan pada Tabel 2.

Tabel 2. Spesifikasi Lingkungan Pengujian

Komponen	Spesifikasi
Perangkat	ASUS TUF Gaming A15
Prosesor	AMD Ryzen 5 7535HS
RAM	16 GB
Sistem Operasi	Windows 11 Home
Browser	Brave v 1.92.134
Server	Dokploy
Tools Pengujian	Google Lighthouse v10.0.0

2.4. Rancangan Pengujian

Pengujian fungsional menggunakan pendekatan *black box* yang memeriksa kesesuaian *output* berdasarkan masukan tanpa mengevaluasi struktur kode *internal* [13]. Selenium WebDriver digunakan untuk mengotomatisasi interaksi pengguna, seperti mengisi formulir, memilih jadwal, menekan tombol, dan memverifikasi pesan atau perubahan status. Pengujian ini dipilih agar skenario dapat dijalankan secara konsisten dan berulang [14]. Skenario mencakup registrasi, login, pencarian ketersediaan, pengajuan reservasi, persetujuan, pengelolaan ruangan, validasi bentrok jadwal, instalasi PWA, *offline fallback*, dan pemulihan koneksi.

Pengujian performa menggunakan Google Lighthouse pada halaman beranda, formulir reservasi, riwayat reservasi, dan dashboard admin. Metrik yang diukur meliputi *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Total Blocking Time* (TBT), *Cumulative Layout Shift* (CLS), dan *Speed Index* (SI). FCP dan LCP mengukur kecepatan tampilan awal dan konten utama, TBT menunjukkan durasi halaman tidak responsif, CLS menilai stabilitas visual, sedangkan SI menggambarkan kecepatan kemunculan konten secara keseluruhan [16], [17]. Setiap kombinasi halaman dan skenario diuji tiga kali menggunakan skenario pada Tabel 3, kemudian nilai rata-ratanya dianalisis.

Tabel 3. Skenario Pengujian Performa

No	Skenario	Kondisi Jaringan	*Service Worker*	*Cache Browser*	Prosedur

1	PWA *cache offline*	*Offline*	Terpasang	Tersedia	Aplikasi dibuka terlebih dahulu saat online hingga seluruh aset tersimpan, kemudian jaringan diubah menjadi *offline*.
2	PWA *cache online*	*Online*	Terpasang	Tersedia	Aplikasi telah diakses sebelumnya sehingga aset telah tersimpan pada *cache*.
3	PWA *no cache offline*	*Offline*	Terpasang	Kosong	Site Data dihapus melalui Chrome DevTools sehingga *cache* tidak tersedia, kemudian jaringan diubah menjadi *offline*.
4	PWA *no cache online*	*Online*	Terpasang	Kosong	Site Data dibersihkan sebelum akses pertama aplikasi.
5	Non-PWA *cache online*	*Online*	Tidak terpasang	Tersedia	Halaman telah diakses sebelumnya sehingga browser menyimpan cache HTTP.
6	Non-PWA *no cache online*	*Online*	Tidak terpasang	Kosong	Site Data *browser* dibersihkan sebelum pengujian.

Sumber: Arjanu & Suartana, 2025

Untuk memudahkan interpretasi hasil pengujian, nilai yang diperoleh pada setiap metrik dibandingkan dengan ambang batas yang direkomendasikan oleh Google Lighthouse. Ambang batas tersebut mengelompokkan performa ke dalam kategori good, needs improvement, dan poor sehingga setiap hasil pengukuran dapat dinilai berdasarkan standar yang sama. Kriteria penilaian tersebut disajikan pada Tabel 4.

Tabel 4. Kriteria Penilaian Metrik Google Lighthouse

No	Metrik	Baik (*Good*)	Perlu Perbaikan (*Needs Improvement*)	Buruk (*Poor*)
1	*Largest Contentful Paint* (LCP)	≤ 2,5 s	> 2,5 s – 4,0 s	> 4,0 s
2	*First Contentful Paint* (FCP)	≤ 1,8 s	> 1,8 s – 3,0 s	> 3,0 s
3	*Speed Index* (SI)	≤ 3,4 s	> 3,4 s – 5,8 s	> 5,8 s
4	*Total Blocking Time* (TBT)	≤ 200 ms	> 200 ms – 600 ms	> 600 ms
5	*Cumulative Layout Shift* (CLS)	≤ 0,10	> 0,10 – 0,25	> 0,25

3. Hasil dan Pembahasan

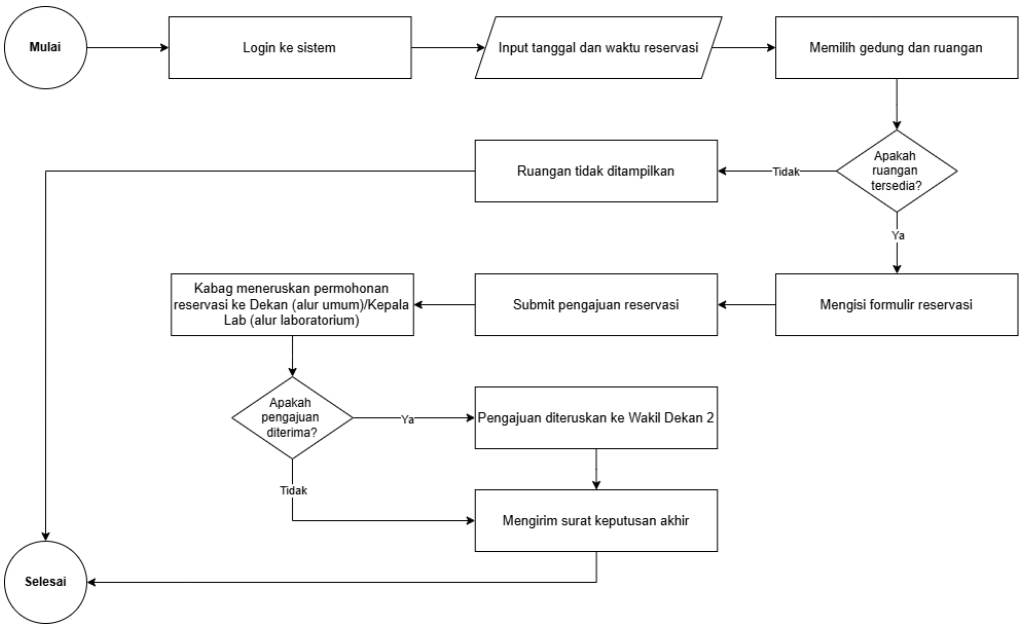
3.1. Implementasi Sistem Reservasi

Hasil pengembangan berupa aplikasi reservasi ruangan berbasis *Progressive Web App* (PWA) yang dapat diakses melalui peramban maupun dipasang pada perangkat pengguna. Sistem menyediakan 4 aktor, yaitu pengguna, *approval* admin, *schedule* admin, dan superadmin yang memiliki hak akses sesuai dengan tugas masing-masing. Pembagian hak akses disajikan pada Tabel 5.

Tabel 5. Fitur Sistem Berdasarkan Aktor

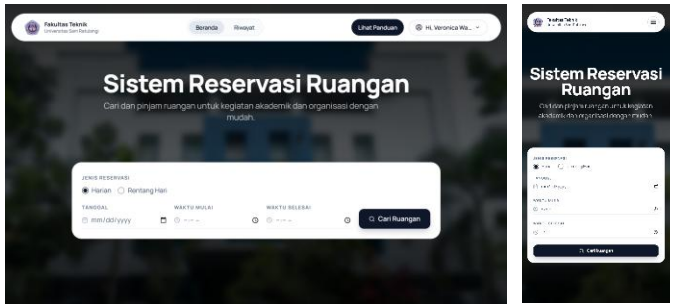
No	Aktor	Fitur Utama
1	Pengguna	Login menggunakan akun UNSRAT, melihat gedung dan ruangan, mengecek ketersediaan jadwal, mengajukan reservasi, mengunggah surat, melihat status dan riwayat reservasi.
2	*Approval* Admin	Melihat *dashboard*, memverifikasi pengajuan sesuai kewenangan, menyetujui atau menolak reservasi.
3	*Schedule* Admin	Melihat *dashboard* dan mengelola jadwal.
4	Superadmin	Memantau seluruh reservasi, mengelola data gedung, ruangan, pengguna, serta *template* surat keputusan.

Alur yang diusulkan dimulai ketika pengguna masuk ke sistem, menentukan jadwal, dan memilih ruangan yang tersedia. Sistem menyaring ruangan berdasarkan data reservasi sehingga slot yang bentrok tidak dapat dipilih. Pengajuan yang valid diteruskan kepada *approval* admin sesuai kewenangan, kemudian diproses hingga menghasilkan keputusan akhir. Alur tersebut mengurangi pemeriksaan jadwal manual dan menyediakan status pengajuan yang dapat dipantau. Gambar memperlihatkan alur proses reservasi yang dikembangkan.



Gambar 2. Alur proses reservasi ruangan yang diusulkan

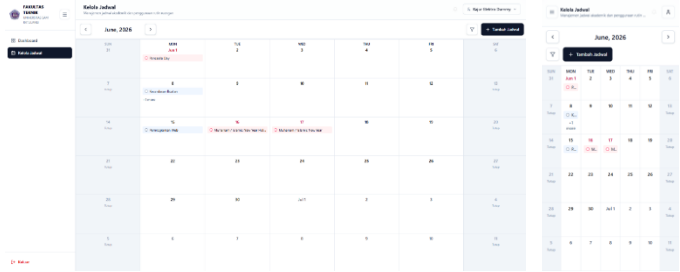
Antarmuka dikembangkan dengan pendekatan responsif. Pada perangkat *desktop*, formulir pencarian dan tabel ditampilkan secara horizontal untuk memanfaatkan ruang layar. Pada perangkat bergerak, komponen disusun vertikal dan tabel tertentu diubah menjadi kartu agar informasi tetap terbaca. Implementasi halaman yang digunakan oleh pengguna pada *desktop* dan *mobile* ditunjukkan pada Gambar. Hasil ini memperlihatkan bahwa satu basis kode dapat melayani berbagai perangkat tanpa memerlukan aplikasi *native* terpisah, sejalan dengan karakteristik PWA.



(a) (b)

Gambar3. Halaman beranda (a) tampilan *desktop* (b) tampilan *mobile* pada aktor pengguna

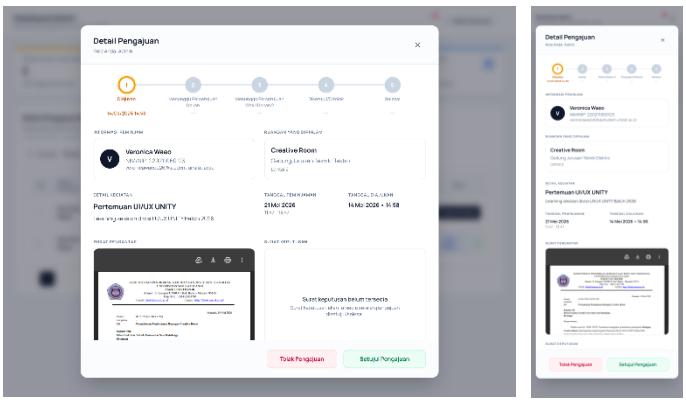
Gambar menunjukkan implementasi halaman kelola jadwal yang digunakan oleh *schedule* admin untuk mengelola jadwal penggunaan ruangan.



(a) (b)

Gambar 4. Halaman kelola jadwal (a) tampilan *desktop* (b) tampilan *mobile* pada aktor *schedule* admin

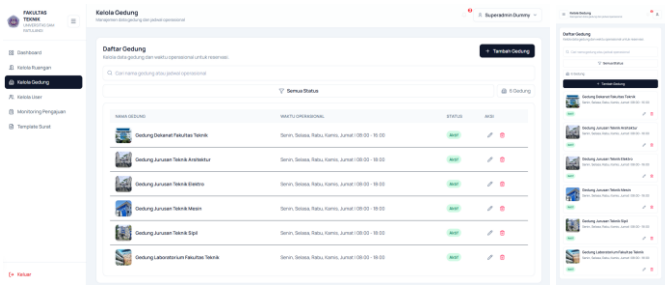
Gambar menunjukkan implementasi halaman detail pengajuan yang digunakan oleh *approval* admin untuk meninjau informasi reservasi serta memberikan keputusan persetujuan atau penolakan.



(a) (b)

Gambar 5. Halaman detail pengajuan (a) tampilan *desktop* (b) tampilan *mobile* pada aktor *approval* admin

Gambar menunjukkan implementasi halaman kelola gedung yang digunakan oleh superadmin untuk mengelola data gedung yang tersedia dalam sistem.

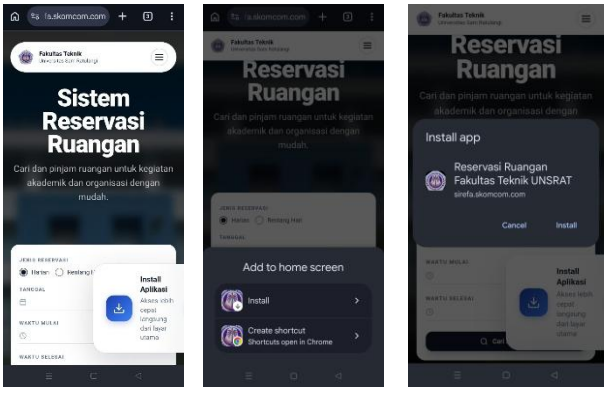


(a) (b)

Gambar 6. Halaman kelola gedung (a) tampilan *desktop* (b) tampilan *mobile* pada aktor superadmin

3.2. Implementasi Fitur PWA

Manifest membuat aplikasi dikenali sebagai PWA dan memungkinkan pengguna memasangnya melalui opsi *Add to Home Screen*. Ketika aplikasi dibuka setelah instalasi, tampilan menggunakan mode *standalone* sehingga menyerupai aplikasi perangkat bergerak. Proses instalasi terdiri atas pemilihan cara instalasi pada aplikasi, konfirmasi melalui peramban, dan persetujuan pemasangan pada perangkat. Rangkaian tersebut ditunjukkan pada Gambar .



(a) (b) (c)

Gambar 7. Proses instalasi PWA (a) tampilan *button install* aplikasi (b) pilihan Add to Home Screen pada *browser* (c) konfirmasi instalasi aplikasi pada perangkat.

Implementasi *web app manifest* dilakukan melalui berkas manifest.json yang mendefinisikan identitas aplikasi, tampilan, ruang lingkup, serta ikon yang digunakan saat aplikasi dipasang pada perangkat. Potongan konfigurasi utama manifest ditunjukkan pada Potongan Kode Program 1.

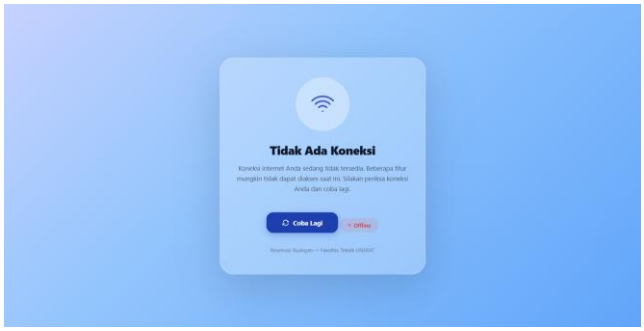
Potongan Kode Program 1. File manifest.json

```
{
  "name": "Reservasi Ruangan Fakultas Teknik UNSRAT",
  "short_name": "Reservasi Fatek",
  "start_url": "/landingpage",
  "display": "standalone",
  "background_color": "#f5f5f0",
  "theme_color": "#1e40af",
  "scope": "/",
  "icons": [...]
}
```

Konfigurasi tersebut menunjukkan bahwa aplikasi menggunakan mode *standalone* sehingga dapat dijalankan menyerupai aplikasi *native* tanpa elemen utama browser. Properti start_url menentukan halaman awal setelah aplikasi dibuka, sedangkan theme_color, background_color, dan icons digunakan untuk menyesuaikan tampilan aplikasi ketika dipasang pada perangkat.

Aplikasi yang telah dipasang tetap menggunakan basis kode dan alamat layanan yang sama dengan versi *browser*. Perbedaannya terlihat pada cara aplikasi diluncurkan, keberadaan ikon pada layar utama, hilangnya sebagian elemen antarmuka *browser*, dan pengalaman navigasi yang lebih menyerupai aplikasi mandiri. Pembaruan tidak memerlukan pemasangan paket baru secara manual. Ketika *service worker* mendeteksi versi aset yang lebih baru, *cache* diperbarui dan versi tersebut digunakan pada akses berikutnya. Mekanisme ini mempermudah distribusi perubahan, tetapi tetap memerlukan pengendalian versi *cache* agar pengguna tidak terlalu lama menerima aset lama.

Service worker menyimpan halaman *fallback*, *manifest*, logo, ikon, dan aset statis utama. Ketika koneksi terputus, halaman yang pernah dimuat masih dapat menampilkan aset dan data terakhir yang tersimpan. Apabila halaman yang diminta belum tersedia dalam *cache*, sistem menampilkan halaman *offline fallback* yang memberikan informasi bahwa koneksi tidak tersedia, seperti pada Gambar .



Gambar 8. Halaman *offline fallback* ketika jaringan tidak tersedia

Strategi *cache* disesuaikan dengan jenis data. Aset statis menggunakan *cache first*, data API menggunakan *network first*, sedangkan halaman navigasi mengutamakan jaringan dengan halaman *fallback* saat koneksi gagal. Pendekatan ini menjaga data dinamis tetap terbaru sekaligus mempercepat pemuatan aset statis melalui *service worker* dan *cache*. Potongan Kode Program 2 menunjukkan proses pendaftaran aset yang akan disimpan ke dalam *cache* saat *service worker* diinstal.

Potongan Kode Program 2. File *service worker* utama PWA bagian *install* dan *precache*

```
const PRE_CACHE_ASSETS = [
  '/offline.html',
  '/manifest.json',
  '/Logo_Fatek_Unsrat.png',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
];

self.addEventListener('install', (event)
=> {
  event.waitUntil(
```

Kemampuan *offline* pada sistem tidak berlaku untuk seluruh fitur dan bersifat terbatas. Tabel 6 menunjukkan fitur yang tetap dapat digunakan serta fitur yang memerlukan koneksi internet.

Tabel 6. Kemampuan Sistem pada Kondisi Offline

No	Fitur	Offline
1	Menampilkan halaman offline fallback	✓
2	Membuka aplikasi	✓
3	Menampilkan halaman yang pernah diakses	✓
4	Melakukan login/register	✗
5	Melihat jadwal terbaru	✗
6	Mengajukan reservasi	✗
7	Menyetujui/menolak reservasi	✗

3.3. Hasil Pengujian Fungsional

Pengujian fungsional dilakukan menggunakan Selenium WebDriver untuk mengotomatisasi proses pengujian terhadap fungsi utama sistem. Sebanyak 17 *test case* dijalankan yang mencakup proses registrasi dan login, pencarian ketersediaan ruangan, reservasi, persetujuan, pengelolaan data, serta fitur khusus *Progressive Web App* (PWA). Setiap pengujian memverifikasi kesesuaian antara hasil aktual dengan hasil yang diharapkan berdasarkan *test case* yang telah dirancang.

Hasil pengujian menunjukkan bahwa 16 dari 17 *test case* berhasil dijalankan sesuai harapan, sedangkan 1 *test case* gagal, yaitu pengujian penghapusan akun pengguna yang telah memiliki riwayat reservasi selesai (UMG-01). Pada skenario tersebut sistem masih menolak penghapusan akun dan menampilkan pesan kesalahan, sedangkan hasil yang diharapkan adalah akun dapat dihapus tanpa menghilangkan riwayat reservasi. Temuan ini menunjukkan bahwa mekanisme pengelolaan relasi data pengguna dan riwayat reservasi masih memerlukan penyempurnaan. *Test case* yang mewakili setiap kategori fitur disajikan pada

Tabel **7**.

Tabel 7. *Test Case* Tiap Kategori

Test ID	*Precondition*	*Test Steps*	*Expected Result*	*Actual Result*	*Status*
REG-01	Belum memiliki akun	Isi seluruh *form* registrasi dengan data valid lalu klik Daftar	Sistem mengirim OTP ke email dan mengarahkan ke halaman verifikasi OTP	Sesuai harapan	*Passed*
RES-02	Ruangan sudah di-*booking* user lain	User lain mencoba mengajukan reservasi pada jadwal yang sama	Sistem menampilkan pesan *conflict* reservasi	Sesuai harapan	*Passed*
APR-01	Reservasi berstatus PENDING	Admin menyetujui pengajuan reservasi	Status reservasi berubah ke tahap *approval* berikutnya	Sesuai harapan	*Passed*
UMG-01	Superadmin sudah login dan terdapat akun user yang memiliki riwayat reservasi berstatus selesai	Hapus akun user pada menu Kelola User	Akun user berhasil dihapus dan riwayat reservasi tetap tersimpan	Sistem menolak penghapusan dan menampilkan pesan *error*	*Failed*
PWA-02	Dalam keadaan *offline*	Instalasi, *fallback offline*, dan pemulihan koneksi berjalan	Non-PWA: Menampilkan halaman error bawaan *browser* "No Internet" PWA: Menampilkan halaman *offline fallback*	Sesuai harapan	*Passed*

Secara *keseluruhan*, hasil pengujian *black box* pada seluruh kategori pengujian disajikan pada Tabel 8.

Tabel 8. Ringkasan Hasil Pengujian *Black Box*

Kategori	Jumlah *Test Case*	Hasil	Status
Registrasi dan login	4	Kredensial dan *domain* email tervalidasi	*All Passed*
Ketersediaan dan reservasi	4	Ruangan tersedia ditampilkan dan konflik dicegah	*All Passed*
Persetujuan	2	Status mengikuti alur persetujuan/penolakan	*All Passed*
Kelola data	4	Tambah data, proteksi riwayat, namun gagal menghapus pengguna	*3 Passed, 1 Failed*
Fitur PWA	3	Instalasi, *fallback offline*, dan pemulihan koneksi berjalan	*All Passed*

3.4. Hasil Pengujian Performa

Pengujian performa dilakukan menggunakan Google Lighthouse pada empat halaman utama, yaitu halaman beranda, formulir reservasi, riwayat reservasi, dan *dashboard* admin. Setiap halaman diuji pada enam skenario sesuai rancangan pengujian dan diulang sebanyak tiga kali. Parameter yang diamati meliputi *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Total Blocking Time* (TBT), *Cumulative Layout Shift* (CLS), *Speed Index* (SI), serta skor performa keseluruhan. Hasil pengujian masing-masing halaman disajikan pada Tabel 9 hingga Tabel 12.

Tabel 9. Hasil Pengujian Performa Halaman Beranda

No	Skenario	Sistem	FCP	LCP	TBT	CLS	SI	Skor Performa
1	*Cache offline*	PWA	0.3	0.3	20	0	0.5	100
2	*Cache online*	PWA	0.3	0.3	20	0.013	0.4	100
3	*No cache offline*	PWA	0.6	0.6	0	0	0.7	100
4	*No cache online*	PWA	0.3	0.9	50	0.013	0.4	99
5	*Cache online*	Non-PWA	0.5	0.6	20	0	1.4	99
6	*No cache online*	Non-PWA	0.8	1.3	30	0	1.3	96

Tabel 10. Hasil Pengujian Performa Halaman Formulir Reservasi

No	Skenario	Sistem	FCP	LCP	TBT	CLS	SI	Skor Performa
1	*Cache offline*	PWA	0.3	0.3	30	0	0.5	100
2	*Cache online*	PWA	0.3	0.4	20	0	0.5	100
3	*No cache offline*	PWA	0.3	0.8	30	0	1.8	96
4	*No cache online*	PWA	0.4	0.9	90	0	3.0	91
5	*Cache online*	Non-PWA	0.3	0.4	70	0	0.7	100
6	*No cache online*	Non-PWA	0.4	0.9	90	0	3.0	91

Tabel 11. Hasil Pengujian Performa Halaman Riwayat Reservasi

No	Skenario	Sistem	FCP	LCP	TBT	CLS	SI	Skor Performa
1	*Cache offline*	PWA	0.3	0.3	30	0	0.4	100
2	*Cache online*	PWA	0.3	0.3	20	0.008	0.4	100
3	*No cache offline*	PWA	0.3	0.5	30	0	1.5	98
4	*No cache online*	PWA	0.5	1.3	30	0.001	1.6	94
5	*Cache online*	Non-PWA	0.3	0.3	20	0	0.3	100
6	*No cache online*	Non-PWA	0.4	0.6	30	0	1.3	99

Tabel 12. Hasil Pengujian Performa Halaman *Dashboard* Admin

No	Skenario	Sistem	FCP	LCP	TBT	CLS	SI	Skor Performa
1	*Cache offline*	PWA	0.3	0.3	20	0	0.5	100
2	*Cache online*	PWA	0.3	0.3	10	0	0.6	100
3	*No cache offline*	PWA	0.4	0.5	30	0	0.8	100
4	*No cache online*	PWA	0.3	0.3	10	0	0.6	100
5	*Cache online*	Non-PWA	0.3	0.3	80	0	0.4	100
6	*No cache online*	Non-PWA	0.4	0.6	30	0	1.3	99

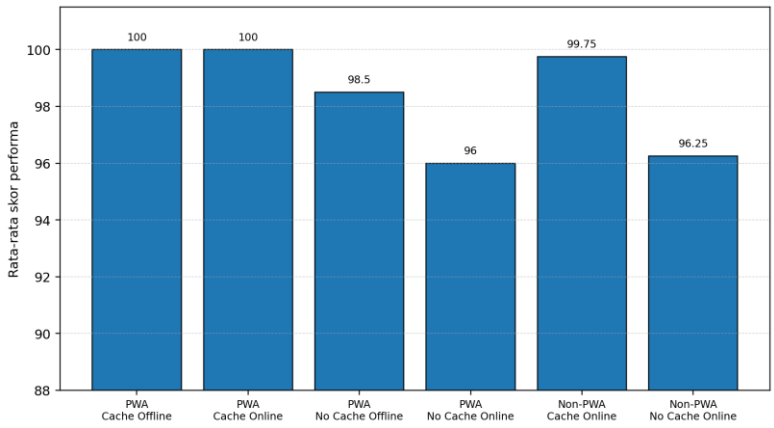
Perbedaan performa antarhalaman dipengaruhi oleh karakteristik konten dan proses pemuatan. Halaman beranda dan *dashboard* admin memperoleh performa lebih stabil karena didominasi aset statis,

sedangkan formulir reservasi dan riwayat reservasi memuat lebih banyak data dinamis dan komponen interaktif sehingga skor pada akses awal tanpa *cache* cenderung lebih rendah.

Perhitungan rata-rata per skenario menunjukkan *cache offline* dan *cache online* PWA sama-sama memperoleh skor maksimal 100. *No cache offline* PWA memperoleh 98,5 dan *no cache online* PWA memperoleh 96. Non-PWA memperoleh 99,75 pada *cache online* dan 96,25 pada *no cache online*. Ringkasan hasil pengujian disajikan dalam Tabel 13 dan divisualisasikan pada Gambar .

Tabel 13. Rata-rata Skor Performa Setiap Skenario

No	Skenario	Sistem	Skor
1	Cache offline	PWA	100
2	Cache online	PWA	100
3	No cache offline	PWA	98,5
4	No cache online	PWA	96
5	Cache online	Non-PWA	99,75
6	No cache online	Non-PWA	96,25



Gambar 9. Perbandingan rata-rata skor performa PWA dan non-PWA

Nilai tinggi pada skenario *cache* menunjukkan bahwa penyimpanan aset statis dapat mengurangi pengambilan sumber daya dari jaringan. Pada kondisi *online* tanpa *cache*, skor PWA dan non-PWA relatif berdekatan, yaitu 96 dan 96,25, sehingga komponen PWA tidak memberikan penalti performa yang berarti. Keunggulan PWA lebih terlihat pada ketersediaan akses karena versi non-PWA tidak dapat digunakan saat *offline*, sedangkan PWA masih mampu menampilkan aset yang tersimpan atau halaman *fallback*. Sebagian besar metrik juga berada dalam kategori baik, dengan FCP dan LCP umumnya di bawah 1,3 detik, TBT sebesar 0–90 milidetik, serta CLS mendekati 0.

Perbandingan kedua versi menunjukkan bahwa PWA tidak selalu memperoleh skor *online* yang lebih tinggi pada setiap halaman karena terdapat proses tambahan, seperti registrasi *service worker*. Namun, perbedaannya relatif kecil, sedangkan PWA menyediakan kemampuan instalasi, *cache*, dan respons saat jaringan terputus yang tidak tersedia pada versi non-PWA. Meskipun demikian, hasil pengujian fungsional dan performa menunjukkan bahwa penerapan PWA layak digunakan sebagai dasar digitalisasi sistem reservasi ruangan di Fakultas Teknik UNSRAT.

4. Kesimpulan

Penelitian ini berhasil mengembangkan sistem reservasi ruangan berbasis *Progressive Web App* (PWA) untuk mendukung proses reservasi di Fakultas Teknik Universitas Sam Ratulangi. Sistem menyediakan fitur pencarian ketersediaan ruangan, pengajuan reservasi, pemantauan status, persetujuan, serta pengelolaan gedung dan ruangan. Hasil pengujian fungsional menunjukkan 16 dari 17 *test case* berjalan sesuai dengan hasil yang diharapkan. Pengujian performa menggunakan Google Lighthouse menunjukkan bahwa PWA memperoleh rata-rata skor 100 pada kondisi *cache offline* dan *cache online*, 98,5 pada *no cache offline*, serta 96 pada *no cache online*. Sementara itu, versi non-PWA

memperoleh skor 99,75 pada *cache online* dan 96,25 pada *no cache online*, namun tidak mendukung akses *offline*. Hasil tersebut menunjukkan bahwa implementasi PWA mampu menyediakan kemampuan instalasi dan akses *offline* terbatas tanpa memberikan penurunan performa *online* yang signifikan dibandingkan versi non-PWA.

Pengembangan selanjutnya dapat dilakukan melalui integrasi sistem dengan layanan akademik di lingkungan Universitas Sam Ratulangi, pengujian pada variasi perangkat dan kondisi jaringan yang lebih beragam, *load testing*, serta *usability testing* untuk memperoleh gambaran yang lebih komprehensif terhadap kualitas sistem.

Daftar Pustaka

[1] F. Sinlae, F. S. Rosyad, F. Nurhidayat, and W. Jannah, "Evolusi Teknologi Web dan Dampaknya Terhadap Masyarakat Digital", Jurnal Ilmu Multidisiplin, vol. 3, pp. 146-154, 2024, doi: 10.38035/jim.v3i2.608.

[2] T. Wahyudi, "Pengembangan Aplikasi Berbasis Web dan Android Sebagai Penunjang Kerja di Indonesia: Systematic Literature Review", Indonesian Journal Computer Science, vol. 1, no. 2, pp. 96-102, 2022.

[3] T. E. Kotambunan and T. Witono, "Sistem Manajemen Peminjaman Ruangan untuk Kegiatan Kemahasiswaan Berbasis Website di Universitas Kristen Maranatha", Jurnal STRATEGI, vol. 7, no. 1, pp. 109-121, 2025.

[4] F. A. Pangestu, F. Athallah, A. Sofwan, and Y. Christyono, "Pengembangan Aplikasi Sistem Informasi Reservasi Ruangan Berbasis Website", Transient: Jurnal Ilmiah Teknik Elektro, vol. 13, no. 3, pp. 115-124, 2024, doi: 10.14710/transient.v13i3.115-124.

[5] R. A. Saputra, I. A. Kautsar, N. Ariyanti, and C. Taurusta, "*Implementation Of Progressive Web Apps On Digital Freelance Platform With Rapid Application Development Method*", SMATIKA: STIKI Informatika Jurnal, vol. 14, no. 1, pp. 157-168, 2024, doi: 10.32664/smatika.v14i01.1228.

[6] G. Matiini, R. Setiyadi, A. Setiawan, and M. Ramli, "Pengembangan Aplikasi Progressive Web Application (PWA) untuk Pembelajaran dan Evaluasi Kelas English Grammar Online Course", JPE (Jurnal Pendidikan Edutama), vol. 8, no. 2, pp. 163-176, 2021.

[7] A. J. Thomas and S. R. Kumar, "*A Study on Progressive Web Apps: Revolutionizing User Experiences and Redefining Web Applications*", ShodhKosh: Journal of Visual and Performing Arts, vol. 5, no. 6, 2024, doi: 10.29121/shodhkosh.v5.i6.2024.5977.

[8] R. J. P. Joarno, M. Fajar, and A. Yunus, "Implementasi *Progressive Web Apps* pada Website GetHelp Menggunakan Next.js", Jurnal KHARISMA Tech, vol. 17, no. 2, 2022.

[9] A. Muawwal, "*The Implementation of PWA (Progressive Web App) Technology in Enhancing Website Performance and Mobile Accessibility*", Buletin Pos dan Telekomunikasi, vol. 22, no. 1, pp. 25-36, 2024, doi: 10.17933/bpostel.v22i1.395.

[10] V. Jagatha, A. Khamesipour, and S. Chung, "*Cross-Platform App Development: A Comparative Study of PWAs and React Native Mobile Apps*", Proceedings of the Information Systems Education Conference, vol. 9, 2023.

[11] M. S. K. Umar and A. A. M. Achmad, "Pengembangan Sistem Reservasi Ruang Laboratorium Berbasis Laravel 11 dengan Fitur Kalender Interaktif di STIKES Bhakti Husada Mulia Madiun", Jurnal Ilmiah Penelitian Mahasiswa, vol. 3, no. 5, pp. 377-383, 2025, doi: 10.61722/jipm.v3i5.1450.

[12] H. R. Hasibuan, A. Supriatman, and C. R. Hidayat, "Sistem Informasi Reservasi Layanan Psikolog dengan Metode Rapid Application Development", Jurnal Informatika dan Teknik Elektro Terapan, vol. 12, no. 3, 2024, doi: 10.23960/jitet.v12i3.4411.

[13] N. M. D. Febriyanti, A. A. K. O. Sudana, and I. N. Piarsa, "Implementasi *Black Box Testing* pada Sistem Informasi Manajemen Dosen", JITTER: Jurnal Ilmiah Teknologi dan Komputer, vol. 2, no. 3, pp. 535-544, 2021, doi: 10.24843/JTRTI.2021.v02.i03.p12.

[14] A. Budiman, M. N. Rahman, I. R. Raul, R. Taufiqurrohman, and A. Saifudin, "Efektivitas Selenium dalam Pengujian Fungsionalitas Aplikasi Kasir Berbasis Web dengan Metode Blackbox", JRIIN: Jurnal Riset Informatika dan Inovasi, vol. 1, no. 1, pp. 252-261, 2023.

[15] I. Armaini, M. H. Dar, and B. Bangun, "*Evaluation of Labuhanbatu Regency Government Website Based on Performance Variables*", Sinkron: Jurnal dan Penelitian Teknik Informatika, vol. 7, no. 2, pp. 760-766, 2022, doi: 10.33395/sinkron.v7i2.11404.

[16] N. Cahyono, U. A. Saputro, and M. F. Salim, "Perbandingan Efektivitas Metode Minifikasi Kode dalam Meningkatkan *Speed Index* dan *Largest Contentful Paint*", JATI (Jurnal Mahasiswa Teknik Informatika), vol. 8, no. 6, 2024, doi: 10.36040/jati.v8i6.11669.

[17] M. R. Wayahdi and F. Ruziq, "Pemodelan Sistem Penerimaan Anggota Baru dengan *Unified Modeling Language* (UML) ", Jurnal Minfo Polgan, vol. 12, no. 1, pp. 1514-1521, 2023, doi: 10.33395/jmp.v12i1.12870.

[18] G. S. G. González, E. R. R. Zurita, and L. A. G. López, "*Progressive Web Applications (PWAs): Architecture, Functioning, and Applications*", Journal of Software Engineering and Computing, vol. 1, no. 2, 2026, doi: 10.5281/zenodo.18985652.