

Dashboard Web Real-Time untuk Monitoring
Sistem Audio Multi-Zona Masjid

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



AHMAD NUR ROHIM
22EO10037

PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026

Dashboard Web Real-Time untuk Monitoring
Sistem Audio Multi-Zona Masjid

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



AHMAD NUR ROHIM
22EO10037

PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa penulisan skripsi dengan judul “*Dashboard Web Real-Time untuk Monitoring Sistem Audio Multi-Zona Masjid*” adalah hasil karya saya dengan arahan dari pembimbing dan belum diajukan kepada pihak manapun. Sumber informasi yang dikutip dalam skripsi ini telah dicantumkan dalam Daftar Pustaka.

Demikian pernyataan ini dibuat dengan sebenarnya. Apabila di kemudian hari terdapat ketidaksesuaian dalam pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.



Cilacap, 25 Juli 2026

A handwritten signature in black ink, appearing to read 'Ahmad Nur Rohim'.

Ahmad Nur Rohim
22EO10037

LEMBAR PENGESAHAN

Skripsi Saudara

Nama : Ahmad Nur Rohim
NIM : 22FO10037
Fakultas/Prodi : Fakultas Matematika dan Ilmu Komputer (FMiKOM)/
Informatika
Judul : *Dashboard Web Real-Time* untuk *Monitoring Sistem*
Audio Multi-Zona Masjid

Telah disidangkan oleh Dewan Penguji Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap pada hari / tanggal :

Rabu , 1 Juli 2026

Dan dapat diterima sebagai pemenuhan tugas akhir mahasiswa Program Strata Satu
Program Studi Informatika (S.Kom.) Fakultas Matematika dan Ilmu Komputer
(FMiKOM) pada Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap.

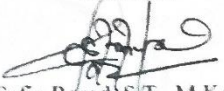
Cilacap, 14 Juli 2026

Dewan Sidang

Ketua


M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

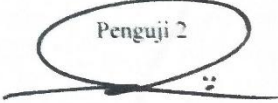
Sekretaris


Safiq Rosad, S.T., M.Kom.
NIDN. 0609018101

Penguji 1


M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

Penguji 2


Tri Anggoro, M.Kom.
NIDN. 0601128103

Pembimbing I


Safiq Rosad, S.T., M.Kom.
NIDN. 0609018101

Pembimbing II


Koko Edy Yulianto, M.Kom.
NIP. 230714144

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Komputer


Mochamad T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

HALAMAN NOTA KONSULTAN

Mochamad T.A. Aziz Zein, S.Si., M.Kom.

Dosen Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap

NOTA KONSULTAN

Hal : Skripsi Saudara Ahmad Nur Rohim
Lampiran : -

Kepada
Yth. Dekan Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap
di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah membaca, memeriksa dan melakukan perbaikan seperlunya maka skripsi saudara/i:

Nama : Ahmad Nur Rohim
NIM : 22EO10037
Prodi : Fakultas Matematika dan Ilmu Komputer (FMIKOM)/ Informatika
Judul : *Dashboard Web Real-Time* untuk *Monitoring* Sistem Audio Multi-Zona Masjid

Dapat diajukan ke Fakultas Matematika dan Ilmu Komputer, Universitas Nahdlatul Ulama Al Ghazali Cilacap untuk memenuhi syarat memperoleh gelar Strata Satu (S1).

Wassalamu'alaikum Wr. Wb.

Cilacap, 15 Juli 2026
Konsultan


Mochamad T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

NOTA PEMBIMBING

Cilacap, 25 Juni 2026

Kepada Yth :

Fakultas Matematika dan Komputer (FMIKOM)

UNUGHA Cilacap

di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah melakukan bimbingan, telaah, arahan dan koreksi tahap penulisan skripsi saudara:

Nama : Ahmad Nur Rohim
NIM : 22EO10037
Fakultas : Ilmu Komputer
Prodi : Fakultas Matematika dan Ilmu Komputer (FMIKOM)/ Informatika
Judul : *Dashboard Web Real-Time untuk Monitoring Sistem Audio Multi-Zona Masjid*

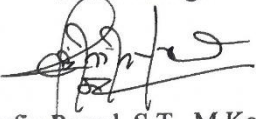
Kami berpendapat bahwa skripsi tersebut sudah dapat diajukan ke sidang skripsi. Bersamaan ini kami kirimkan skripsi tersebut, semoga dapat segera disidangkan.

Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum Wr. Wb.

Mengetahui,

Pembimbing I


Safiq Rosad, S.T., M.Kom.
NIDN. 0609018101

Pembimbing II


Koko Edy Yulianto, M.Kom.
NIP. 230714144

HALAMAN MOTO

“Dan jika kamu membalas, maka balaslah dengan (balasan) yang sama dengan siksaan yang ditimpakan kepadamu. Tetapi jika kamu bersabar, sesungguhnya itulah yang lebih baik bagi orang yang sabar”

(QS. An-Nahl : 126)

Pendidikan merupakan perlengkapan paling baik untuk hari tua. (*Aristoteles*)

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT, Rabb semesta alam yang senantiasa memberikan karunia sehingga penulis mampu menyelesaikan penulisan skripsi ini. Karya ini saya persembahkan kepada :

1. Orang tua Bapak Usman Tohir dan Ibu Maryatun Khariah yang selalu mendidik saya, memberikan do'a, dukungan, nasihat dan semangat yang tiada henti.
2. Kakak tercinta Ahmad Nur Rohman dan adik Ayu Azzahro Ahmad yang selalu memberikan do'a dan semangat tiada henti di setiap detik langkahku.
3. Keluarga FMIKOM Angkatan 2022 yang selalu memberikan keceriaan, kebersamaan dan motivasi.
4. Keluarga program studi Informatika, Sistem Informasi, Matematika yang saya banggakan
5. Seluruh teman UNUGHA yang telah memberikan do'a, dukungan, dan semangat.

KATA PENGANTAR

Puji syukur kepada Allah SWT atas segala limpahan nikmat dan karuniaNya, sehingga penulis dapat menyelesaikan penulisan skripsi. Sholawat dan salam senantiasa penulis haturkan kepada Nabi Muhammad SAW sebagai pembimbing seluruh umat manusia.

Skripsi ini tidak akan selesai tanpa adanya bantuan dari banyak pihak, karena itu penulis menyampaikan terima kasih kepada :

1. Bapak M.T Abdul Aziz Zein, M.Kom., Dekan FMIKOM UNUGHA.
2. Bapak Safiq Rosad, S.T., M.Kom., pembimbing I yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
3. Bapak Koko Edy Yulianto, M.Kom., pembimbing II yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
4. Bapak Verry, S.T., M.Kom., sebagai Pembimbing Akademis.
5. Instansi SHATOMEDIA sebagai pelaksanaan skripsi.
6. Bapak-Ibu dosen Program Studi FMIKOM UNUGHA.
7. Semua pihak yang tidak dapat kami sebutkan satu persatu.

Semoga Allah SWT membalas jerih payah dan pengorbanan yang telah diberikan dengan balasan yang lebih baik. Amiin. Penulis berharap semoga karya kecil ini bermanfaat bagi pembaca.

Cilacap, 25 Juni 2026



Penulis

Ahmad Nur rohim

ABSTRAK

Ahmad Nur Rohim. *Dashboard Web Real-Time Untuk Monitoring Sistem Audio Multi-Zona Masjid*. Dibimbing oleh Safiq Rosad, S.T., M.Kom. dan Koko Edy Yulianto, M.Kom.

Sistem distribusi audio multi-zona di masjid yang dikembangkan dengan *arsitektur Cyber-Physical System (CPS)* memerlukan antarmuka kontrol yang digunakan agar pengelola masjid dapat melakukan *monitoring* kondisi sistem secara *real-time*. Penelitian ini mengembangkan *dashboard web* yang terintegrasi dengan *backend* sistem melalui *REST API* dan *WebSocket* untuk mendukung pemantauan zona audio secara terpusat. Penelitian menggunakan pendekatan *Research and Development (R&D)* dengan model pengembangan *Prototype*, serta berfokus pada pengembangan *dashboard web* yang dirancang sesuai kebutuhan pengelola masjid sebagai pengguna utama.

Metodologi penelitian meliputi: (1) analisis kebutuhan melalui wawancara dengan pengelola masjid; (2) perancangan antarmuka sistem menggunakan *wireframe* dan *mockup*; (3) implementasi *dashboard web* menggunakan teknologi *frontend* dan *backend* dengan integrasi *REST API* dan *WebSocket* untuk komunikasi data *real-time*; (4) pengujian fungsional dan kinerja sistem.

Kontribusi penelitian ini mencakup: (1) pengembangan *dashboard web* untuk *monitoring* sistem audio multi-zona masjid yang mendukung pemantauan kondisi zona secara *real-time* bagi admin, teknisi, dan operator; (2) implementasi sistem *monitoring real-time* berbasis *REST API* dan *WebSocket*; (3) evaluasi sistem *dashboard* pada konteks *Smart Mosque* melalui pengujian fungsional dan pengujian kinerja untuk memastikan keandalan dan performa sistem.

Hasil penelitian menunjukkan bahwa *dashboard web* berhasil dikembangkan untuk mendukung *monitoring* sistem audio masjid multi-zona secara *real-time* melalui integrasi *REST API* dan *WebSocket*. Pengujian fungsional menunjukkan seluruh fitur utama berjalan sesuai kebutuhan sistem, sedangkan pengujian kinerja menunjukkan komunikasi *real-time* berlangsung dengan latensi rendah dan stabil. Dengan demikian, *dashboard* yang dikembangkan dapat digunakan sebagai sarana *monitoring* terpusat pada sistem audio masjid multi-zona berbasis *Cyber-Physical System (CPS)*.

Kata kunci: Dashboard Web, REST API, WebSocket, Cyber-Physical System, Smart Mosque, Sistem Audio Multi-Zona

ABSTRACT

Ahmad Nur Rohim. Real-Time Web Dashboard for Monitoring a Mosque's Multi-Zone Audio System. Supervised by Safiq Rosad, S.T., M.Kom. and Koko Edy Yulianto, M.Kom.

The multi-zone audio distribution system in a mosque, developed using a Cyber-Physical System (CPS) architecture, requires a control interface that enables mosque administrators to monitor the system's condition in real time. This study develops a web dashboard integrated with the system backend through REST API and WebSockets to support centralized monitoring of audio zones. The research employs a Research and Development (R&D) approach using the Prototype development model and focuses on developing a web dashboard designed according to the needs of mosque administrators as the primary users.

The research methodology includes: (1) needs analysis through interviews with mosque managers; (2) system interface design using wireframes and mockups; (3) web dashboard implementation using front-end and back-end technologies with REST API and WebSocket integration for real-time data communication; and (4) functional and performance testing of the system.

The contributions of this research include: (1) development of a web dashboard for monitoring a multi-zone mosque audio system that supports real-time monitoring of zone conditions for administrators, technicians, and operators; (2) implementation of a real-time monitoring system based on REST API and WebSocket; and (3) evaluation of the dashboard system in the context of a Smart Mosque through functional and performance testing to ensure system reliability and performance.

The results of this research indicate that the web dashboard was successfully developed to support real-time monitoring of a multi-zone mosque audio system through the integration of REST API and WebSocket. Functional testing demonstrated that all key features functioned according to system requirements, while performance testing demonstrated that real-time communication occurred with low latency and stability. Therefore, the developed dashboard can be used as a centralized monitoring tool for a multi-zone mosque audio system based on a Cyber-Physical System (CPS).

Keywords: *Web Dashboard, REST API, WebSocket, Cyber-Physical System, Smart Mosque, Multi-Zone Audio System*

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI	Error! Bookmark not defined.
LEMBAR PENGESAHAN	Error! Bookmark not defined.
HALAMAN NOTA KONSULTAN	iv
NOTA PEMBIMBING	Error! Bookmark not defined.
HALAMAN MOTO	vii
HALAMAN PERSEMBAHAN	viii
KATA PENGANTAR	Error! Bookmark not defined.
ABSTRAK	ix
<i>ABSTRACT</i>	xi
DAFTAR ISI	xii
DAFTAR GAMBAR	xiv
DAFTAR TABEL	xv
BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Rumusan Masalah	2
C. Batasan Masalah	2
D. Tujuan Penelitian	3
E. Manfaat Penelitian	4
F. Kontribusi Penelitian	4
G. Keaslian Penelitian	5
H. Keterbaruan penelitian	5
BAB II TINJAUAN PUSTAKA	7
A. Penelitian Terkait	7
B. Landasan Teori	16
1. Sistem Audio Multi-Zona di Masjid	16
2. Dasar Teori Dashboard Web	17
3. Teknologi Web Waktu Nyata	17
4. Kerangka Kerja React dan Pola Kinerja	18
5. Pustaka Visualisasi Data	19

BAB III METODOLOGI.....	20
A. Jenis Penelitian	20
B. Waktu dan Tempat Penelitian	20
C. Subjek dan Objek Penelitian	21
D. Alat dan Bahan	22
E. Prosedur Penelitian.....	22
1. Tahap Analisis Kebutuhan.....	23
2. Tahap Perancangan (Desain)	24
3. Tahap Implementasi.....	32
4. Tahap Pengujian	32
5. Tahap Evaluasi dan Perbaikan.....	36
F. Jadwal Penelitian.....	36
BAB IV HASIL DAN PEMBAHASAN.....	38
A. Hasil Implementasi Dashboard Web	38
1. Deskripsi Implementasi Sistem	38
2. Hasil Pengujian Fungsional Dashboard.....	42
3. Hasil Pengujian Kinerja Dashboard.....	45
4. Hasil Pengujian Konektivitas <i>Real-time</i>	47
B. Analisis Keunggulan dan Keterbatasan.....	47
1. Keunggulan Dashboard.....	47
2. Keterbatasan dan Rekomendasi Perbaikan	48
C. Ringkasan Hasil Pengujian.....	49
BAB V KESIMPULAN.....	51
A. Kesimpulan.....	51
B. Saran/Rekomendasi	51
DAFTAR PUSTAKA	53
LAMPIRAN.....	56
Lampiran 1	56
Lampiran 2	57
Lampiran 3	59
Lampiran 4	61
Lampiran 5	63
Lampiran 6	66
Lampiran 7	69

Lampiran 8	69
Lampiran 9	73

DAFTAR GAMBAR

Gambar 2. 1 Diagram mind map.....	15
Gambar 2. 2 Posisi Penelitian	16
Gambar 3. 1 Diagram Alur Metodologi Penelitian.....	23
Gambar 3. 2 Arsitektur Sederhana Dashboard Web Sistem Audio Masjid	24
Gambar 3. 3 Use Case Diagram	25
Gambar 3. 4 Activity Diagram Akses dan Monitoring Dashboard Audio Masjid	26
Gambar 3. 5 Sequence Diagram Login & Monitoring Dashboard Audio	27
Gambar 3. 6 Halaman Login.....	28
Gambar 3. 7 Halaman Ringkasan Zona	29
Gambar 3. 8 Halaman Ringkasan Detail Zona	29
Gambar 3. 9 Halaman Detail dari Zona	30
Gambar 3. 10 Halaman Grafik Historis	30
Gambar 3. 11 Alur Data WebSocket pada Dashboard Web	31
Gambar 3. 12 Tahap Implementasi Dashboard Web	32
Gambar 3. 13 Tahap Evaluasi dan Perbaikan Dashboard Web	36
Gambar 4. 1 Arsitektur Sistem Dashboard Web Tiga Lapis.....	38
Gambar 4. 2 Technology Stack - Sistem Dashboard Audio Masjid	39
Gambar 4. 3 Dashboard Overview.....	41
Gambar 4. 4 Mockup Halaman Detail Zona dengan Grafik SPL 24 Jam.....	42
Gambar 5. 1 Dokumentasi bersama Narasumber Terkait	56
Gambar 5. 2 Dokumentasi Pengujian Kompatibilitas Device	57
Gambar 5. 3 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)	57
Gambar 5. 4 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)	58
Gambar 5. 5 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)	58
Gambar 5. 6 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)	58
Gambar 5. 7 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)	59
Gambar 5. 8 Dokumentasi Pengujian Halaman Detail Zona 1	59
Gambar 5. 9 Dokumentasi Pengujian Halaman Detail Zona 2	60
Gambar 5. 10 Dokumentasi Pengujian Halaman Detail Zona 3	60
Gambar 5. 11 Dokumentasi Pengujian Halaman Detail Zona 4	60
Gambar 5. 12 Dokumentasi Pengujian Halaman Detail Zona 5	61
Gambar 5. 13 Dokumentasi Page Load Dashboard Zona	62
Gambar 5. 14 Dokumentasi Page Load Dashboard Detail Zona	62
Gambar 5. 15 Dokumentasi Page Load Dashboard Detail Perzona.....	63
Gambar 5. 16 Dokumentasi Page Load Dashboard Grafik Historis	63

DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu	7
Tabel 2. 2 Gap Penelitian dan Novelty	11
Tabel 4. 1 Spesifikasi Teknologi.....	40
Tabel 4. 2 Hasil Pengujian Fitur Dashboard	42
Tabel 4. 3 Hasil Pengujian Real-Time Update Dashboard	43
Tabel 4. 4 Hasil Pengujian Kompatibilitas Device	44
Tabel 4. 5 Hasil Pengujian Page Load Time.....	45
Tabel 4. 6 Hasil Pengujian Resource Usage	46
Tabel 4. 7 Hasil Pengujian WebSocket Latency	47
Tabel 4. 8 Keterbatasan dan Rekomendasi Perbaikan	48
Tabel 4. 9 Ringkasan Pencapaian Pengujian Dashboard Web.....	49
Tabel 5. 1 Dokumentasi hasil dari tahap pengujian Rata-Rata Page Load Time dan Kinerja Testing Dashboard	61
Tabel 5. 2 Dokumentasi hasil dari tahap pengujian fungsional dashboard Detail Zona.....	63
Tabel 5. 3 Dokumentasi hasil dari tahap pengujian fungsional dashboard grafik historis Filter dan Export Data	65
Tabel 5. 4 Checklist Persiapan Lingkungan Pengujian Dashboard Web.....	66
Tabel 5. 5 Checklist Setup Perangkat Client.....	67
Tabel 5. 6 Checklist Setup Lingkungan Simulasi	67
Tabel 5. 7 Daftar Akun Pengujian Dashboard	68
Tabel 5. 8 Ringkasan Data Awal Database	69
Tabel 5. 9 Checklist Verifikasi Data Awal	69
Tabel 5. 10 Daftar Endpoint REST API yang Digunakan Dashboard.....	69

BAB I PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi informasi mendorong pengelolaan sistem audio masjid tidak hanya bergantung pada pengaturan manual, tetapi mulai mengarah pada pemantauan berbasis sistem informasi terintegrasi[1]. Pada masjid berukuran menengah hingga besar, sistem audio umumnya dibagi ke dalam beberapa zona agar distribusi suara dapat disesuaikan dengan fungsi setiap area, seperti ruang utama, serambi, atau area luar masjid. Semakin banyak zona yang digunakan, semakin besar pula kebutuhan untuk memantau kondisi setiap zona secara bersamaan. Tanpa sistem *monitoring* yang terpusat, pengurus harus melakukan pengecekan satu per satu sehingga gangguan, seperti speaker tidak aktif, koneksi perangkat terputus, atau perubahan nilai *SPL*, berpotensi terlambat diketahui dan dapat memengaruhi kualitas penyampaian suara kepada jamaah[2].

Dalam praktiknya, beberapa masjid sudah menggunakan aplikasi bergerak (*mobile*) sederhana untuk membantu admin (*Administrator*), teknisi (*Maintenance*) dan operator masjid melakukan pengaturan dasar, seperti mengatur volume atau menyalakan dan mematikan zona tertentu[3][4]. Namun, kebutuhan admin, teknisi dan operator lebih kompleks, misalnya memantau status seluruh zona sekaligus, menganalisis riwayat gangguan, serta meninjau tren kualitas audio dari waktu ke waktu. Kebutuhan ini kurang terpenuhi jika hanya mengandalkan aplikasi bergerak dengan tampilan layar terbatas.

Penggunaan *dashboard web* untuk *monitoring* sistem audio masjid menampilkan layar yang lebih luas, dukungan perangkat masukan seperti papan ketik dan *tetikus*, serta kemampuan menampilkan banyak komponen *visual analitis* dalam satu tampilan[5]. Dengan *dashboard monitoring* bagi admin, teknisi dan operator dapat memantau beberapa zona audio secara bersamaan, melihat status zona (aktif, tidak aktif, atau bermasalah), serta mengakses grafik historis tanpa perlu berpindah layar terlalu sering[6][7].

Meskipun beberapa penelitian telah membahas penggunaan sistem audio berbasis *Public Address (PA)* di lingkungan masjid serta permasalahan pengelolannya, dan sebagian lainnya mengkaji integrasi sistem berbasis waktu nyata menggunakan *WebSocket* pada sistem terdistribusi, namun belum banyak penelitian yang secara khusus mengembangkan *dashboard web* terpusat untuk *monitoring* sistem audio masjid multi-zona yang mendukung visualisasi historis dan pembaruan data *real-time* secara simultan[8][9]. Kesenjangan ini menunjukkan

perlunya pengembangan sistem dashboard berbasis web yang mampu mengintegrasikan pemantauan multi-zona, analisis historis, serta komunikasi waktu nyata dalam satu platform terpusat.

Integrasi *dashboard web* dengan sistem fisik cerdas *CPS (cyber-physical system)* pada masjid memungkinkan data dari perangkat tepi (node audio dan pengendali zona) dikirim ke peladen belakang (*backend*). Data tersebut kemudian diakses oleh dashboard melalui *REST API* sebagai media pertukaran data antara *backend* dan aplikasi *klien*, serta *WebSocket* untuk pembaruan data secara *real-time*[9][8][10]. Penggunaan *REST API* menjadi penting karena memungkinkan pengelolaan data yang terstruktur, konsisten, serta mudah diintegrasikan dengan berbagai komponen sistem. Sementara itu, *WebSocket* mendukung penyajian informasi secara cepat dengan latensi rendah sehingga perubahan kondisi sistem dapat segera ditampilkan pada *dashboard*[11]. Kondisi ini menjadi urgensi dalam pengembangan *dashboard web monitoring* sistem audio multi-zona, karena pengelola masjid membutuhkan informasi dari seluruh zona secara terpusat, akurat, dan responsif terhadap perubahan kondisi sistem secara *real-time*.

Berdasarkan uraian tersebut, perlu dikembangkan suatu sistem berbasis *REST API* dan *WebSocket* untuk *dashboard web* sebagai alat *monitoring* sistem audio multi-zona masjid[12]. Sistem tersebut dirancang untuk *monitoring* kondisi zona audio secara *real-time*. *Dashboard* ini diharapkan dapat membantu pengelola masjid memantau kondisi multi-zona secara *real-time*, menganalisis data historis, serta meningkatkan kecepatan respons ketika terjadi gangguan audio.

B. Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini dapat dirumuskan sebagai berikut.

1. Bagaimana merancang *dashboard Web* yang dapat menampilkan 8–16 zona audio secara *real-time* dengan tampilan yang dapat digunakan oleh admin, teknisi dan operator masjid?
2. Bagaimana mengintegrasikan *dashboard web* dengan *REST API backend* dan *WebSocket* untuk mendukung komunikasi data *real-time* dengan keterlambatan (latensi) yang rendah?
3. Bagaimana mengoptimalkan kinerja *dashboard Web* agar mampu menampilkan dan memperbarui data dalam jumlah cukup besar, seperti data titik pengukuran suara dan status zona per hari?
4. Bagaimana mengevaluasi fungsionalitas, kinerja, dan responsivitas *dashboard web* untuk memastikan sistem dapat berjalan dengan baik pada perangkat desktop, laptop, dan tablet?

C. Batasan Masalah

Untuk menjaga fokus penelitian ini, beberapa batasan masalah yang ditetapkan adalah:

1. Pengembangan *frontend* hanya menggunakan kerangka kerja *React* versi 18.2.0 dengan bahasa pemrograman *TypeScript*, dan tidak membahas implementasi dengan kerangka kerja *web* lain.
2. Mekanisme komunikasi waktu nyata antara peladen dan *dashboard Web* menggunakan *WebSocket*, sehingga pendekatan lain seperti *Server-Sent Events* atau long polling tidak dibahas secara mendalam.
3. Jumlah zona audio yang ditangani maksimal 16 zona, disesuaikan dengan skenario sistem audio multi-zona yang menjadi acuannya tanpa membahas pengembangan skalabilitas zona secara mendalam.
4. Data *historis* yang ditampilkan pada dashboard dibatasi pada rentang waktu maksimal 24 jam terakhir, sedangkan pengelolaan data *historis* jangka panjang di luar cakupan penelitian ini.
5. Penelitian tidak membahas teknik visualisasi tingkat lanjut seperti visualisasi tiga dimensi atau pemodelan spasial suara, serta tidak mencakup penerapan pembelajaran mesin untuk prediksi kualitas audio.
6. Pengujian sistem difokuskan pada pengujian fungsional menggunakan *Black Box Testing* dan pengujian kinerja menggunakan *Chrome DevTools*, *React DevTools*, *Lighthouse*, serta *monitoring Docker* pada peramban modern yang umum digunakan.
7. Penelitian ini difokuskan pada pengembangan *dashboard web* untuk *monitoring* sistem audio masjid multi-zona secara *real-time* menggunakan *REST API* dan *WebSocket*. Penelitian tidak mencakup pengembangan fitur analisis prediktif berbasis *Machine Learning* maupun deteksi otomatis terhadap potensi kerusakan perangkat.
8. Penelitian ini hanya membahas implementasi *monitoring* melalui *dashboard web*, sehingga integrasi dengan layanan notifikasi eksternal, seperti WhatsApp, Telegram, maupun Email, tidak termasuk dalam ruang lingkup penelitian.

D. Tujuan Penelitian

Tujuan penelitian ini disusun untuk menjawab rumusan masalah yang telah dijabarkan.

1. Mengembangkan sebuah *dashboard web* untuk *monitoring* sistem audio masjid multi-zona yang dapat menampilkan hingga 8–16 zona audio secara bersamaan.
2. Mewujudkan integrasi antara *dashboard web*, *REST API backend*, dan *WebSocket* sehingga pembaruan data kondisi zona audio dapat berlangsung secara waktu nyata dengan latensi di bawah batas yang ditetapkan.
3. Mengoptimalkan kinerja tampilan dan pemrosesan data pada *dashboard web* sehingga tetap responsif ketika menampilkan beberapa grafik dan pembaruan data secara langsung.

4. Melakukan evaluasi *dashboard web* melalui pengujian fungsional, pengujian kinerja, dan pengujian kompatibilitas perangkat untuk memastikan sistem berjalan dengan baik pada desktop, laptop, dan tablet.

E. Manfaat Penelitian

Penelitian ini diharapkan memberikan beberapa manfaat, baik secara praktis maupun akademis.

1. Bagi pengelola masjid
Menyediakan alat bantu pemantauan sistem audio multi-zona yang lebih terstruktur dan *informatif*. Membantu mempercepat identifikasi dan penanganan masalah audio melalui tampilan status dan peringatan pada *dashboard web*.
2. Bagi pengembang sistem dan mahasiswa
Menjadi contoh penerapan pengembangan *dashboard web* waktu nyata yang terintegrasi dengan sistem fisik pada lingkungan masjid. Menambah pengalaman dalam merancang antarmuka admin, teknisi dan operator berbasis web yang menggabungkan pemantauan dan visualisasi data.
3. Bagi pengembangan ilmu pengetahuan
Memberikan kontribusi berupa studi kasus penerapan teknologi *web real-time* dan *dashboard monitoring* bagi admin, teknisi dan operator pada domain sistem audio masjid. Menambah referensi terkait perancangan antarmuka admin teknisi dan operator untuk sistem multi-zona, khususnya pada konteks rumah ibadah.

F. Kontribusi Penelitian

Kontribusi penelitian ini diarahkan agar relevan dengan kebutuhan praktis di lapangan dan sejalan dengan perkembangan penelitian terkait sistem audio masjid dan masjid pintar.

1. Menghasilkan rancangan dan prototipe *dashboard web* yang dapat digunakan sebagai antarmuka admin, teknisi dan operator untuk *monitoring* zona sistem audio masjid secara waktu nyata.
2. Menyediakan pola integrasi sederhana antara *dashboard web* dengan layanan *REST API* dan *WebSocket* yang dapat dijadikan acuan pada pengembangan sistem sejenis di lingkungan masjid atau sistem *Internet of Things* lainnya.
3. Menyajikan hasil pengujian sistem *dashboard* meliputi pengujian fungsional, performa halaman, latensi komunikasi *WebSocket*, serta

penggunaan sumber sebagai evaluasi awal terhadap kinerja *dashboard web* yang dikembangkan.

4. Menambahkan contoh implementasi *dashboard monitoring* bagi admin, teknisi dan operator untuk sistem multi-zona di ranah masjid yang dapat dimanfaatkan sebagai rujukan bagi penelitian atau tugas akhir berikutnya.

G. Keaslian Penelitian

Penelitian mengenai sistem audio masjid, sistem *monitoring* berbasis *web*, serta komunikasi data menggunakan *REST API* dan *WebSocket* telah banyak dilakukan pada berbagai domain. Namun, berdasarkan hasil kajian penelitian terdahulu, belum ditemukan penelitian yang secara khusus mengembangkan *dashboard web* untuk *monitoring* sistem audio multi-zona masjid yang mengintegrasikan *REST API* dan *WebSocket* dalam satu sistem *monitoring real-time*.

Penelitian ini dikembangkan dengan mengacu pada penelitian-penelitian sebelumnya sebagai landasan teori dan teknologi yang digunakan, namun memiliki objek, implementasi, serta ruang lingkup yang berbeda. Oleh karena itu, penelitian ini merupakan hasil pengembangan yang memiliki karakteristik tersendiri sesuai kebutuhan *monitoring* sistem audio multi-zona masjid.

H. Keterbaruan penelitian

Penelitian tugas akhir ini memiliki keterbaruan pada penerapan *dashboard web* yang dirancang khusus untuk *monitoring* sistem audio masjid multi-zona. *Dashboard* yang dikembangkan mampu menampilkan kondisi zona secara *real-time*, menyajikan data historis *SPL*, serta mendukung proses evaluasi performa sistem melalui pengujian fungsional dan kinerja *dashboard*. Dengan demikian, penelitian ini tidak hanya berfokus pada aspek teknis sistem audio, tetapi juga pada penyediaan antarmuka *monitoring* yang sesuai dengan kebutuhan operasional pengelolaan audio masjid.

Keterbaruan (novelty) penelitian ini dapat dilihat dari beberapa aspek berikut.

1. Fokus domain sistem audio masjid multi-zona

Penelitian ini secara spesifik mengkaji pengembangan *dashboard web* untuk *monitoring* sistem audio masjid multi-zona. Berbeda dengan penelitian *monitoring* pada domain umum seperti lingkungan, energi, dan industri, penelitian ini berfokus pada kebutuhan pemantauan kondisi zona audio masjid yang memerlukan penyajian informasi secara terpusat untuk mendukung operasional sistem audio.

2. Integrasi *dashboard web* dengan *REST API* dan *WebSocket* untuk *monitoring* zona audio

Meskipun *REST API* dan *WebSocket* telah banyak digunakan pada sistem *monitoring* modern, penelitian ini mengintegrasikan kedua teknologi tersebut pada *dashboard monitoring* sistem audio masjid multi-zona. *REST API* digunakan untuk menyediakan data konfigurasi, detail zona, dan data historis, sedangkan *WebSocket* digunakan untuk menyampaikan perubahan status dan nilai *SPL* secara *real-time* tanpa memerlukan pemuatan ulang halaman.

3. Penggabungan pemantauan *real-time* dan pengujian performa sistem

Penelitian ini tidak hanya menghasilkan prototipe *dashboard web* untuk *monitoring* sistem audio masjid multi-zona, tetapi juga melakukan pengujian performa sistem yang meliputi waktu muat halaman, latensi komunikasi *WebSocket*, penggunaan sumber daya sistem, kompatibilitas perangkat, serta respons layanan *REST API*. Pengujian tersebut memberikan gambaran mengenai kinerja *dashboard* yang dikembangkan dalam mendukung *monitoring* sistem audio secara *real-time*.

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Penelitian mengenai sistem audio masjid multi-zona menunjukkan bahwa pengaturan zona dan tata letak pengeras suara sangat mempengaruhi pemerataan tingkat tekanan suara dan kenyamanan jamaah. Kajian sistematis tentang distribusi audio di masjid menyoroti pentingnya pembagian zona, pengaturan waktu tunda, serta pemanfaatan pemrosesan sinyal untuk meningkatkan kejelasan suara, tetapi belum banyak membahas antarmuka dashboard khusus bagi admin, teknisi dan operator.

Tabel 2. 1 Penelitian Terdahulu

No.	Penulis & Tahun	Keterangan
(1)	(2)	(3)
1	Handojo et al., 2025	Judul: <i>Integrating Real-Time IoT Based Monitoring and Dashboard Design for Closed-House Hen Farming.</i> Metode: <i>Research and Development (R&D), implementasi Internet of Things (IoT), dan evaluasi sistem monitoring real-time.</i> Tools: ESP32, sensor IoT, Dashboard Web, Firebase, Internet of Things (IoT). Hasil: <i>Sistem monitoring real-time berbasis MQTT berhasil berjalan baik.</i>
2	Mabotha et al., 2025	Judul: <i>A Dockerized Approach to Dynamic Endpoint Management for RESTful Application Programming Interfaces in Internet of Things Ecosystems.</i> Metode: Eksperimen dan Evaluasi Framework IoT. Tools: Docker, RESTful API, PostgreSQL, Dynamic Endpoint Framework. Hasil: <i>Framework REST API dinamis mampu mempertahankan throughput tinggi, latensi rendah, dan mendukung skalabilitas sistem IoT.</i>
3	Hitimana et al., 2022	Judul: <i>Containerized Architecture Performance Analysis for IoT Framework Based on Enhanced Fire Prevention Case Study: Rwanda.</i>

		Metode: Eksperimen dan Analisis Performa Arsitektur IoT. Tools: Docker Container, Microservices, Raspberry Pi, Sensor Network, DevOps, IoT Framework. Hasil: Arsitektur <i>containerized</i> meningkatkan efisiensi sumber daya dan skalabilitas <i>framework IoT</i> pada <i>smart building</i>.
4	Rosad, Lasimin et al., 2025	Judul: <i>Edge-Based IoT EndPoint Performance Optimization for Smart Mosques</i>. Metode: Eksperimen dan optimasi IoT berbasis edge. Tools: Edge Computing, IoT Endpoint, ESP32, Edge Server, WebSocket. Hasil: <i>Edge computing</i> meningkatkan performa <i>IoT</i> (latency turun, availability naik).
5	Rosad, Yudhana, et al., 2025	Judul: <i>Zoned Audio Distribution: Systematic Review and Optimization Framework</i>. Metode: Systematic Literature Review (SLR). Tools: ODEON, Ease Focus, DSP, STI, RT60. Hasil: Zonasi audio meningkatkan pemerataan dan kejelasan suara.
6	Setiawan et al., 2024	Judul: Rancang bangun sistem informasi <i>monitoring</i> bimbingan tugas akhir berbasis web menggunakan <i>framework reactjs</i>. Metode: Waterfall, UML, ERD, Black Box Testing. Tools: ReactJS, REST API, Database. Hasil: Sistem <i>monitoring ReactJS</i> berjalan baik dan diterima pengguna.
7	Borowiec et al., 2023	Judul: <i>Knowledge Extraction and Discovery about Web System Based on the Benchmark Application of Online Stock Trading System</i>. Metode: Benchmark Testing dan Workload Analysis. Tools: Docker Swarm, Spring Boot, RabbitMQ, MySQL. Hasil: Menganalisis performa sistem web melalui <i>monitoring</i> trafik, CPU,

		RAM, dan perilaku pengguna berdasarkan log sistem.
8	Chodorek & Chodorek, 2025	Judul: <i>Web Real-Time Communications-Based Unmanned-Aerial Vehicle-Borne Internet of Things and Stringent Time Sensitivity: A Case Study.</i> Metode: Eksperimen lapangan, pengukuran latensi, studi komparatif WebRTC vs WebSocket dalam sistem UAV-IoT real-time. Tools: WebRTC dan WebSocket. Hasil: Delay stabil di bawah 10 ms dan <i>WebRTC</i> lebih unggul dari <i>WebSocket</i>.
9	Eka Putra et al., 2024	Judul: Analisis Komparasi Protokol <i>Websocket</i> dan <i>MQTT</i> Dalam Proses <i>Push Notification</i>. Metode: Metode eksperimental dengan pengukuran delay, response time, dan CPU usage. Tools: WebSocket, MQTT, Python, Flask. Hasil: <i>WebSocket</i> lebih responsif dan efisien dibanding <i>MQTT</i>.
10	Fauzi & Amarudin, 2025	Judul: Integrasi <i>Internet of Things</i> dan <i>Web</i> untuk <i>Monitoring</i> Kendali Irigasi Tates Secara <i>Real Time</i>. Metode: Prototyping & Eksperimen Tools: ESP32, MQTT, Web Application, MySQL. Hasil: Sistem <i>monitoring</i> dan kendali irigasi berbasis <i>IoT</i> dan <i>web</i> berhasil diimplementasikan secara <i>real-time</i> dengan baik.
11	Rajagukguk & Widianono, 2025	Judul: Perancangan dan Implementasi Sistem <i>Monitoring</i> Mesin Produksi Berbasis Lampu <i>IoT</i> dan <i>Web Dashboard</i> di PT Motor Jaya Indonesia. Metode: Perancangan IoT + pengujian konektivitas + evaluasi performa. Tools: NodeMCU ESP8266, LED RGB, Firebase, Web Dashboard. Hasil: Status mesin <i>real-time</i> & respon lebih cepat.
12	Trinanda Putra et al., 2024	Judul: <i>A Practical Implementation of RESTful API and WebSocket Protocol for Power Efficiency Monitoring: A</i>

		<i>Case Study at Muhammadiyah Orphanage.</i> Metode: Perancangan dan implementasi sistem IoT <i>monitoring</i> daya listrik berbasis RESTful API dan WebSocket. Tools: ESP32, PZEM-004T, REST API, WebSocket, MicroPython, Telegram Bot, OLED Display. Hasil: Sistem <i>monitoring real-time</i> berhasil diimplementasikan dengan rata-rata latensi <i>WebSocket</i> 136,9 ms.
13	Gilang Sumudra et al., 2025	Judul: Rancang Bangun <i>Website Terpadu Menggunakan Framework ReactJS</i> untuk Meningkatkan Tampilan yang Menarik dan Responsif pada Masjid Jami' At-Taqwa. Metode: Waterfall, Black Box, Unit Testing, UAT. Tools: ReactJS, NodeJS, REST API, MySQL. Hasil: <i>Website</i> terpadu berjalan stabil dengan kepuasan pengguna tinggi.
14	Merdekawan & Sari, 2022	Judul: Studi Awal Rancang Bangun <i>Indoor Farming Monitoring System</i> Berbasis IoT dengan Protokol <i>Websocket</i>. Metode: Rekayasa sistem Waterfall dan analisis delay Framework. Tools: ESP8266, WebSocket, NodeJS, ReactJS. Hasil: <i>WebSocket</i> mendukung <i>monitoring IoT</i> dengan delay rendah.
15	Kojansow et al., 2024	Judul: <i>Internet of Things (IoT) Platform Development using WebSocket Communication.</i> Metode: Research & Development (R&D). Tools: REST API, WebSocket, PostgreSQL, MongoDB. Hasil: Platform <i>IoT</i> berbasis <i>web</i> berjalan <i>real-time</i> dan stabil dengan komunikasi data efisien.
16	Pradana et al., 2026	Judul: <i>Design and Implementation of a React.js-Based Monitoring Dashboard for Tangerang Weather Radar and Server Room Air Conditioning Control.</i> Metode: Implementasi dashboard web. Tools: React.js, Node.js, REST API.

		Hasil: Sistem <i>IoT</i> mampu memantau lingkungan secara <i>real-time</i> .
17	Supriyono & Haekal Noval Akbar, 2025	Judul: Integrasi <i>RESTful API</i> Berbasis JSON untuk Sistem Informasi Manajemen Produk Mitra Secara <i>Real-Time</i> . Metode: <i>Waterfall</i> , <i>RESTful API</i> dan <i>WebSocket</i> . Tools: Laravel, FrankenPHP, WebSocket, JWT, Postman. Hasil: Integrasi <i>REST API</i> dan <i>WebSocket</i> meningkatkan respons <i>real-time</i> .
18	Pamungkas et al., 2025	Judul: Pengembangan Aplikasi Pendaftaran Siswa Baru Berbasis <i>Web</i> Dengan <i>React.js</i> dan <i>Tailwind CSS</i> . Metode: Development Research, Black Box Testing. Tools: ReactJS, Tailwind CSS, REST API. Hasil: Sistem <i>web</i> meningkatkan efisiensi proses administrasi.
19	Rezki et al., 2024	Judul: Sistem Informasi Pengelolaan Masjid Jami' Al-Khoir Berbasis <i>Web</i> . Metode: SDLC, Waterfall. Tools: Laravel, MySQL. Hasil: Sistem informasi masjid meningkatkan efisiensi dan transparansi.
20	Pradipta & Susetyo, 2025	Judul: Pengembangan <i>Dashboard</i> Admin menggunakan <i>React JS</i> dan <i>Ant Design</i> pada Toko Berkas. Metode: SDLC model Waterfall dan pengujian performa. Tools: ReactJS, Ant Design, REST API, AWS. Hasil: <i>Dashboard React</i> memiliki performa dan respons yang baik.
21	Rosad, Yudhana, et al., 2025	Judul: <i>Initial Study of Public Address Usage in Mosque Audio Systems</i> . Metode: Deskriptif kualitatif. Tools: Kuesioner, analisis statistik deskriptif. Hasil: Penggunaan <i>PA system</i> tinggi, namun kepuasan pengguna rendah.

Tabel 2. 2 Gap Penelitian dan Novelty

No	Penulis Lengkap	Fokus Penelitian	Gap Penelitian	Novelty Penelitian
1	Handojo et al., 2025	Pengembangan sistem <i>monitoring IoT</i> berbasis <i>dashboard web real-time</i> .	Belum membahas <i>monitoring</i> sistem audio masjid multi-zona, integrasi <i>REST API</i> dan <i>WebSocket</i> .	Implementasi <i>dashboard web monitoring</i> sistem audio masjid multi-zona berbasis <i>React</i> dan <i>TypeScript</i> yang mengintegrasikan <i>REST API</i> dan <i>WebSocket</i> .
2	Mabotha et al., 2025	Pengembangan <i>Dynamic REST API</i> berbasis <i>Docker</i> untuk sistem <i>IoT</i> .	Belum membahas <i>dashboard monitoring</i> dan visualisasi data <i>real-time</i> .	Implementasi <i>dashboard monitoring</i> audio multi-zona yang memanfaatkan <i>REST API</i> dan <i>WebSocket</i> pada lingkungan <i>Smart Mosque</i> .
3	Hitimana et al., 2022	Arsitektur <i>IoT</i> berbasis <i>microservices</i> dan <i>container Docker Swarm</i> untuk <i>smart building</i> dan <i>fire prevention</i> .	Belum membahas <i>monitoring</i> dan kontrol perangkat audio multi-zona secara <i>real-time</i> menggunakan <i>REST API</i> dan <i>WebSocket</i> .	Implementasi <i>dashboard monitoring</i> audio multi-zona berbasis <i>REST API</i> dan <i>WebSocket</i> pada lingkungan <i>Smart Mosque</i> .
4	Rosad, Lasimin et al., 2025	Optimasi <i>endpoint IoT</i> berbasis <i>edge computing</i> .	Tidak menyediakan <i>dashboard monitoring web</i> .	<i>Dashboard web</i> terintegrasi dengan sistem <i>edge IoT</i> .
5	Rosad, Yudhana, et al., 2025	Optimalisasi <i>zoned</i> audio distribution.	Konseptual, belum diimplementasikan dalam <i>dashboard real-time</i> .	Implementasi <i>monitoring</i> zona audio masjid berbasis <i>web</i> .
6	Setiawan et al., 2024	<i>Monitoring</i> tugas akhir berbasis <i>ReactJS</i> .	Tidak berbasis <i>IoT/CPS</i> , tidak <i>WebSocket</i> .	<i>Monitoring CPS</i> audio berbasis <i>WebSocket</i> .
7	Borowiec et al., 2023	Analisis performa dan <i>workload</i> sistem <i>web</i> berbasis <i>cloud</i> .	Tidak membahas <i>monitoring</i> audio masjid multi-zona.	<i>Dashboard monitoring</i> audio masjid multi-zona <i>real-time</i> berbasis <i>REST API</i> dan <i>WebSocket</i>
8	Chodorek & Chodorek, 2025	Komunikasi <i>real-time</i>	Tidak membahas <i>dashboard</i>	Implementasi <i>WebSocket</i> pada

		<i>WebRTC</i> dan <i>WebSocket</i> pada sistem <i>IoT</i> yang sensitif terhadap latensi.	<i>monitoring</i> audio masjid maupun visualisasi data berbasis web.	<i>dashboard monitoring</i> audio multi-zona berbasis web.
9	Eka Putra et al., 2024	Komparasi <i>WebSocket</i> vs <i>MQTT</i> .	Tidak diimplementasikan dalam studi kasus dashboard operasional.	Implementasi <i>WebSocket</i> pada <i>dashboard</i> audio nyata.
10	Fauzi & Amarudin, 2025	<i>Monitoring AQI</i> berbasis <i>IoT</i> + <i>dashboard</i> web.	Terbatas pada lingkungan tanpa analitik lanjutan.	<i>Monitoring</i> lebih komprehensif, skalabel, dan <i>dashboard</i> interaktif.
11	Rajagukguk & Widianono, 2025	<i>Monitoring</i> mesin produksi berbasis <i>IoT</i> + <i>web dashboard</i> .	Hanya indikator status tanpa analisis performa.	Tambahan analisis performa, data historis optimal, dan peningkatan stabilitas sistem.
12	Trinanda Putra et al., 2024	<i>Monitoring</i> daya listrik <i>real-time</i> berbasis <i>REST API</i> dan <i>WebSocket</i> .	Fokus pada <i>monitoring</i> energi, bukan audio multi-zona masjid.	<i>Monitoring</i> zona audio masjid berbasis <i>dashboard web real-time</i> .
13	Gilang Sumudra et al., 2025	<i>Website</i> terpadu <i>ReactJS</i> .	Tidak menggunakan <i>WebSocket</i> , tidak <i>CPS</i> .	<i>Dashboard monitoring</i> audio multi-zona berbasis <i>ReactJS</i> , <i>REST API</i> , dan <i>WebSocket</i> pada lingkungan <i>Smart Mosque</i>
14	Merdekawan & Sari, 2022	<i>Monitoring IoT</i> berbasis <i>WebSocket</i> .	Belum ada <i>monitoring dashboard web</i> audio lengkap.	<i>Dashboard web monitoring</i> audio masjid lengkap.
15	Kojansow et al., 2024	<i>Monitoring</i> dan kontrol <i>IoT</i> berbasis web.	Sistem masih umum, belum diterapkan pada audio masjid dan multi-zona dan belum berbasis <i>CPS</i> .	<i>Dashboard monitoring</i> audio masjid berbasis <i>REST API</i> & <i>WebSocket</i> dengan pendekatan <i>CPS</i> .
16	Pradana et al., 2026	<i>Dashboard monitoring</i> berbasis <i>React.js</i> untuk radar	Belum membahas <i>monitoring</i> sistem audio masjid multi-zona	<i>Dashboard monitoring</i> audio masjid multi-zona berbasis <i>React</i> , <i>REST API</i> , dan

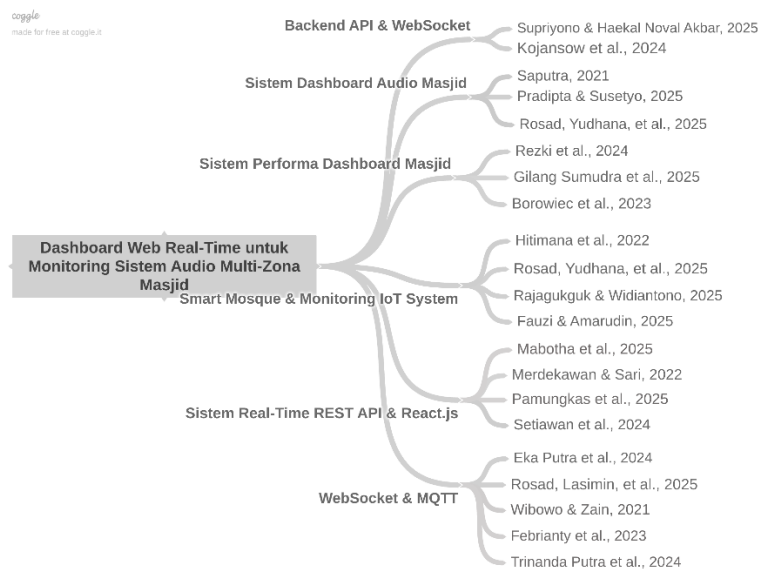
		cuaca dan server <i>room</i> .	berbasis <i>REST API</i> dan <i>WebSocket</i> .	<i>WebSocket</i> untuk pemantauan <i>real-time</i> .
17	Supriyono & Haekal Noval Akbar, 2025	Integrasi <i>REST API</i> + <i>WebSocket</i> .	Fokus <i>backend</i> , tidak ada visualisasi <i>CPS</i> .	<i>Dashboard</i> visualisasi <i>CPS</i> audio <i>real-time</i> .
18	Pamungkas et al., 2025	<i>Aplikasi web</i> berbasis <i>React</i> .	Tidak <i>monitoring IoT</i> dan tidak <i>real-time</i> .	<i>Dashboard monitoring</i> audio multi-zona berbasis <i>ReactJS</i> dengan komunikasi <i>real-time</i> menggunakan <i>WebSocket</i>
19	Rezki et al., 2024	Sistem informasi masjid berbasis <i>web</i> .	Bersifat administratif, tidak <i>monitoring</i> teknis.	<i>Monitoring</i> teknis audio multi-zona.
20	Pradipta & Susetyo, 2025	<i>Dashboard admin React & Ant Design</i> .	Tidak diterapkan pada <i>CPS/IoT</i> .	<i>Dashboard monitoring</i> audio multi-zona berbasis <i>ReactJS</i> dengan integrasi <i>REST API</i> dan <i>WebSocket</i> .
21	Rosad, Yudhana, et al., 2025	Studi penggunaan <i>Public Address (PA)</i> pada sistem audio masjid.	Belum membahas implementasi <i>dashboard web</i> untuk <i>monitoring</i> kondisi sistem audio secara <i>real-time</i> .	Pengembangan <i>dashboard web monitoring</i> audio multi-zona berbasis <i>REST API</i> dan <i>WebSocket</i> untuk mendukung pemantauan operasional masjid.

Penelitian tentang sistem audio masjid multi-zona telah membahas berbagai aspek, seperti distribusi audio, pengaturan zona, dan peningkatan kualitas penyampaian suara kepada jamaah. Namun, berdasarkan hasil kajian penelitian terdahulu, masih terdapat keterbatasan dalam pengembangan *dashboard web* yang mampu melakukan pemantauan sistem audio multi-zona secara *real-time*. Sebagian besar penelitian berfokus pada aspek perangkat keras, arsitektur *IoT*, atau komunikasi data, sementara integrasi *dashboard web* interaktif yang memanfaatkan *REST API* dan *WebSocket* untuk menampilkan status zona audio secara waktu nyata masih belum banyak dibahas. Oleh karena itu, penelitian ini mengusulkan pengembangan *dashboard monitoring* audio masjid multi-zona berbasis *ReactJS* dengan integrasi *REST API* dan *WebSocket* dalam lingkungan *CPS* untuk mendukung pemantauan operasional secara *real-time* oleh admin, teknisi, dan operator.

Kajian mengenai arsitektur masjid pintar berbasis *CPS* menunjukkan adanya integrasi antara perangkat fisik, jaringan komunikasi, dan aplikasi pemantauan dalam satu ekosistem yang saling terhubung. Pendekatan ini memungkinkan proses pengumpulan, pengiriman, dan penyajian data dilakukan secara berkelanjutan sehingga kondisi sistem dapat dipantau secara lebih efektif. Namun, sebagian besar penelitian *CPS* yang telah dilakukan masih berfokus pada aspek integrasi perangkat dan komunikasi data, sementara penerapannya pada sistem *monitoring* audio masjid multi-zona melalui *dashboard web* masih relatif terbatas.

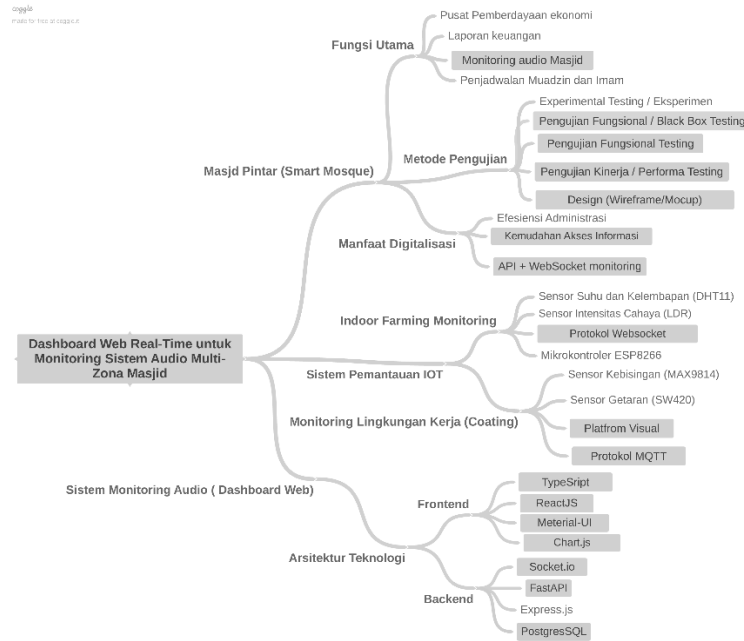
Di sisi lain, perkembangan teknologi *dashboard* berbasis *web* telah memungkinkan penyajian informasi sistem secara lebih interaktif dan mudah dipantau. Berbagai penelitian telah memanfaatkan *framework modern* seperti *ReactJS* serta teknologi komunikasi data *real-time* untuk mendukung kegiatan *monitoring* pada berbagai domain, seperti lingkungan, energi, dan sistem industri. Selain itu, *REST API* dan *WebSocket* banyak digunakan untuk mendukung pertukaran data dan pembaruan informasi secara *real-time* pada *aplikasi web modern*. Meskipun demikian, penerapan kombinasi teknologi tersebut pada sistem audio masjid multi-zona masih relatif terbatas dan belum banyak dievaluasi dari aspek fungsional maupun kinerja sistem.

Berdasarkan kondisi tersebut, masih terdapat celah penelitian dalam pengembangan dan pengujian *dashboard web* untuk *monitoring* sistem audio masjid multi-zona berbasis *REST API* dan *WebSocket*. Penelitian ini berfokus pada: (1) penyajian 8–16 zona audio dalam satu antarmuka terpusat, (2) penyajian data *real-time* dan historis bagi admin, teknisi, dan operator masjid, serta (3) evaluasi performa *dashboard* melalui pengujian fungsional dan pengujian kinerja sistem.



Gambar 2. 1 Diagram mind map

Gambar 2.1 terlihat berdasarkan kajian penelitian terdahulu, pengembangan sistem informasi masjid, *smart mosque* berbasis *IoT*, dan *dashboard* berbasis *web* telah banyak dilakukan, namun masih belum terintegrasi untuk *monitoring* audio masjid multi-zona secara *real-time*. Oleh karena itu, mind map berikut disusun untuk menunjukkan posisi penelitian ini dengan mengintegrasikan backend *FastAPI*, *WebSocket*, *dashboard web* interaktif, serta sistem audio masjid berbasis *IoT*.



Gambar 2. 2 Posisi Penelitian

Pada Gambar 2. 2 terlihat bahwa penelitian ini berada pada irisan antara konsep *Smart Mosque* dan Sistem *Monitoring Audio* berbasis *dashboard web*. Penelitian ini mengintegrasikan perangkat sensor dan komunikasi *real-time* menggunakan *WebSocket* dengan dashboard berbasis *ReactJS* serta backend *FastAPI* dan *PostgreSQL*. Dengan demikian, penelitian ini memperkuat integrasi *monitoring* perangkat fisik dan pemantauan multi-zona audio masjid dalam satu sistem terpusat yang dievaluasi melalui pengujian fungsional dan pengujian kinerja sistem.

B. Landasan Teori

1. Sistem Audio Multi-Zona di Masjid

Sistem audio masjid merupakan sistem pengeras suara yang digunakan untuk mendukung kegiatan ibadah dan aktivitas keagamaan agar pesan dapat terdengar jelas dan merata[13][14]. Pada masjid berskala menengah hingga besar, sistem audio umumnya menerapkan konsep multi-zona, yaitu pembagian area seperti ruang utama, selasar, lantai dua, ruang belajar, dan area luar ke dalam beberapa zona audio agar pengaturan tingkat suara dapat disesuaikan dengan fungsi ruang dan kondisi kegiatan[2]. Pendekatan zona ini membantu menjaga kejelasan suara di area utama ibadah sekaligus mengurangi kebisingan berlebih pada area yang

tidak memerlukan volume tinggi. Selain itu, pembagian zona memudahkan pengelola dalam mengatur dan memantau kondisi audio pada setiap area sesuai kebutuhan operasional masjid.

Namun, pada implementasi sistem audio terpusat, sering muncul permasalahan berupa sulitnya memantau kondisi setiap zona secara bersamaan, terutama ketika indikator status perangkat dan penguat tersebar di beberapa lokasi[1][6]. Tanpa sistem pemantauan terintegrasi, pengurus masjid masih bergantung pada laporan jamaah atau pemeriksaan manual ketika terjadi gangguan audio.

Penelitian terdahulu merekomendasikan penerapan zonasi audio yang lebih terstruktur serta pemanfaatan teknologi informasi untuk mendukung pengelolaan dan pemantauan sistem audio masjid[13][2]. Dalam konteks tersebut, sistem *monitoring* berbasis *web* dapat digunakan untuk menyajikan informasi kondisi perangkat dan zona audio secara terpusat sehingga memudahkan proses *monitoring* sistem. Oleh karena itu, konsep sistem audio multi-zona menjadi salah satu landasan teoritis yang mendukung pengembangan sistem *monitoring* pada penelitian ini.

2. Dasar Teori Dashboard Web

Dashboard web dapat didefinisikan sebagai antarmuka berbasis peramban yang menyajikan informasi penting secara ringkas dalam bentuk teks, angka, dan visualisasi sehingga pengguna dapat memantau kondisi suatu sistem dan mengambil keputusan dengan cepat[5][6]. Pada konteks sistem audio masjid, *dashboard* bertugas menyajikan informasi status zona, tingkat suara, dan kondisi sistem lainnya secara waktu nyata sehingga memudahkan proses *monitoring* sistem.

Prinsip umum perancangan *dashboard* meliputi pemilihan indikator utama (*key metrics*), pengelompokan informasi yang sejenis, penggunaan kode warna yang konsisten, serta penyajian grafik yang mudah dibaca[5]. Selain itu, tampilan perlu disesuaikan dengan tugas pengguna, misalnya menonjolkan peringatan ketika ada zona bermasalah dan menyediakan akses cepat ke detail zona tertentu.

Berbagai contoh *dashboard* pemantauan sistem (misalnya untuk server atau jaringan) menunjukkan bahwa penggunaan susunan kartu informasi (*cards*), grafik garis, dan tabel ringkas dapat membantu pengguna memahami kondisi sistem secara sekilas. Pola ini dapat diadaptasi untuk keperluan pemantauan sistem audio multi-zona dengan menyesuaikan label, satuan, dan konteks informasi untuk lingkungan masjid.

3. Teknologi Web Waktu Nyata

Teknologi web waktu nyata memungkinkan peladen dan *klien web* saling bertukar data secara berkelanjutan tanpa harus melakukan permintaan berulang secara manual oleh pengguna. Beberapa pendekatan yang umum digunakan

meliputi *WebSocket*, *Server-Sent Events*, dan teknik polling, yang masing-masing memiliki karakteristik latensi dan penggunaan sumber daya yang berbeda.

WebSocket menyediakan saluran komunikasi dua arah yang tetap terbuka antara *klien* dan peladen, sehingga cocok digunakan untuk *web dashboard* yang memerlukan pembaruan data cepat dan sering, seperti pemantauan status perangkat[15][8][16]. Dibandingkan teknik polling tradisional, *WebSocket* cenderung memiliki latensi lebih rendah dan beban permintaan jaringan yang lebih kecil karena tidak perlu mengirim permintaan berulang setiap beberapa detik[9][17].

Dalam implementasi sistem *monitoring* berbasis web, *FastAPI* digunakan sebagai mekanisme komunikasi yang memungkinkan pertukaran data antara *klien* dan peladen. *FastAPI* menyediakan endpoint yang dapat diakses oleh aplikasi klien untuk memperoleh maupun mengirimkan data sesuai kebutuhan sistem. Salah satu bentuk *FastAPI* yang banyak digunakan pada aplikasi web modern adalah *REST API*, yang memanfaatkan protokol *HTTP* melalui mekanisme *request-response* untuk menyediakan data seperti konfigurasi sistem, data historis, dan informasi awal yang dibutuhkan *aplikasi*[10].

Sementara itu, *WebSocket* menyediakan saluran komunikasi dua arah yang memungkinkan peladen mengirimkan pembaruan data secara langsung kepada *klien* secara *real-time* tanpa perlu melakukan permintaan ulang secara berkala. Kombinasi *REST API* dan *WebSocket* banyak digunakan pada sistem *monitoring* modern karena mampu menyediakan informasi yang responsif dengan latensi rendah serta penggunaan sumber daya jaringan yang lebih efisien[8][9]. Dengan demikian, perubahan kondisi zona audio dapat segera terlihat di antarmuka admin, teknisi dan operator, sehingga waktu tanggap terhadap gangguan dapat dipersingkat.

4. Kerangka Kerja React dan Pola Kinerja

React merupakan salah satu kerangka kerja antarmuka pengguna berbasis komponen yang banyak digunakan untuk membangun *aplikasi web* interaktif[18][19][4]. *React* memungkinkan pengembang memecah antarmuka menjadi komponen kecil yang dapat digunakan kembali, memudahkan pengelolaan struktur kode dan pemeliharaan aplikasi[5][19].

Dalam konteks aplikasi yang menampilkan data waktu nyata dan banyak pembaruan, pengelolaan kinerja *React* menjadi penting agar tampilan tetap responsif. Beberapa pola yang umum digunakan antara lain meminimalkan penggambaran ulang komponen yang tidak perlu, mengelola status aplikasi secara terpusat, dan memanfaatkan pemuatan kode secara bertahap untuk mengurangi ukuran awal berkas *aplikasi*.

Pustaka tambahan seperti pengelola status dan pustaka tabel atau grafik dapat membantu menyusun struktur *aplikasi dashboard* dengan lebih teratur. Namun, pada tugas akhir tingkat Strata Satu, fokus dapat diarahkan pada pemahaman konsep dasar penggunaan React untuk membangun halaman dashboard, mengelola status zona audio, dan menampilkan grafik sederhana yang terhubung ke sumber data waktu nyata.

5. Pustaka Visualisasi Data

Visualisasi data pada dashboard berperan penting untuk membantu pengguna memahami tren dan kondisi sistem tanpa harus membaca angka secara rinci[7][6]. Pada pemantauan sistem audio masjid, visualisasi dapat berupa grafik garis tingkat tekanan suara terhadap waktu, indikator status zona, serta diagram sederhana yang menunjukkan kondisi umum sistem.

Berbagai pustaka visualisasi untuk web, seperti *Chart.js*, *D3.js*, dan *ECharts*, menyediakan beragam jenis grafik dan opsi penyesuaian tampilan[20][21]. *Chart.js* dikenal relatif ringan dan cukup untuk kebutuhan grafik dasar, sehingga sesuai digunakan pada tugas akhir tingkat S1 yang memerlukan implementasi visualisasi sederhana namun fungsional.

Pemilihan pustaka visualisasi perlu mempertimbangkan keseimbangan antara kebutuhan fitur dan kompleksitas penggunaan. Untuk penelitian ini, penggunaan grafik garis dan batang sederhana sudah memadai untuk menampilkan informasi historis tingkat suara dan status zona, sehingga pengembang dapat lebih fokus pada integrasi data waktu nyata dan penyusunan tampilan antarmuka yang jelas bagi admin, teknisi dan operator masjid.

BAB III METODOLOGI

A. Jenis Penelitian

Penelitian ini termasuk dalam jenis penelitian dan pengembangan (*Research and Development/R&D*) yang berfokus pada perancangan dan implementasi prototipe *dashboard web* untuk *monitoring* sistem audio masjid multi-zona secara *real-time*. Produk utama yang dikembangkan berupa *dashboard web* yang terhubung dengan layanan backend melalui *REST API* dan *WebSocket* untuk menampilkan informasi kondisi zona audio secara waktu nyata.

Pada tingkat Strata Satu, penelitian dan pengembangan dilakukan dalam skala terbatas dengan menyesuaikan kompleksitas fitur dan ruang lingkup pengujian. Oleh karena itu, prototipe yang dihasilkan lebih menekankan pada implementasi fungsionalitas inti, integrasi komunikasi data *real-time*, serta evaluasi performa sistem melalui pengujian fungsional dan pengujian kinerja *dashboard*.

B. Waktu dan Tempat Penelitian

Waktu dan tempat penelitian merupakan aspek penting yang menjelaskan periode pelaksanaan serta lokasi yang digunakan dalam proses pengembangan dan pengujian *dashboard web monitoring* sistem audio multi-zona masjid. Penentuan waktu dan lokasi penelitian dilakukan untuk memastikan seluruh tahapan penelitian dapat dilaksanakan secara sistematis sesuai dengan metode *Research and Development (R&D)* menggunakan model *Prototype*. Adapun waktu dan tempat pelaksanaan penelitian dijelaskan sebagai berikut.

1. Waktu Penelitian

Penelitian ini dilaksanakan pada bulan Desember 2025 hingga Juni 2026. Tahap awal penelitian dimulai pada bulan Desember 2025 melalui observasi dan diskusi dengan pengelola Masjid Al-Muhajirin untuk mengidentifikasi kebutuhan sistem *monitoring* audio multi-zona. Selanjutnya, proses analisis kebutuhan, perancangan sistem, serta pengembangan *dashboard web* menggunakan *React* dan *TypeScript* dilakukan pada bulan Januari hingga Februari 2026.

Pada bulan Maret hingga Mei 2026 dilakukan implementasi sistem, integrasi *dashboard* dengan backend melalui *REST API* dan *WebSocket*, serta pengujian sistem yang meliputi pengujian fungsional (*Black Box Testing*), pengujian kompatibilitas perangkat, pengujian *page load time*, pengujian *resource usage* menggunakan *Chrome DevTools* dan *React DevTools*, serta pengujian latensi komunikasi *WebSocket*. Tahap akhir penelitian dilakukan pada bulan Juni 2026 melalui analisis hasil pengujian, penyusunan laporan penelitian, dan penyelesaian naskah skripsi.

2. Tempat Penelitian

Penelitian ini dilaksanakan pada tiga lokasi yang memiliki peran berbeda sesuai dengan tahapan penelitian, yaitu pengumpulan kebutuhan sistem, pengembangan dashboard web, dan pengujian sistem.

Lokasi pertama adalah Shatomeia Yogyakarta yang digunakan sebagai tempat perancangan dan pengembangan *dashboard web*. Pada lokasi ini dilakukan analisis kebutuhan sistem, perancangan arsitektur perangkat lunak, perancangan antarmuka pengguna (*UI*), implementasi dashboard web menggunakan *React* dan *TypeScript*, integrasi dengan backend melalui *REST API* dan *WebSocket*, serta penyempurnaan fitur berdasarkan kebutuhan sistem yang telah diidentifikasi.

Lokasi kedua adalah Masjid Al-Muhajirin Sedayu, yang beralamat di Jl. Wates KM.11, Perum Griya Kencana Permai, Sedayu, Bantul, Daerah Istimewa Yogyakarta. Lokasi ini digunakan sebagai tempat observasi dan pengumpulan kebutuhan sistem melalui diskusi dengan pengelola masjid. Kegiatan tersebut bertujuan untuk mengidentifikasi kebutuhan fungsional dan nonfungsional sistem audio multi-zona yang menjadi dasar dalam pengembangan *dashboard monitoring*.

Lokasi ketiga adalah Shatomeia Kesugihan, Kabupaten Cilacap, yang digunakan sebagai tempat pelaksanaan pengujian *dashboard web*. Pada lokasi ini dilakukan pengujian fungsional menggunakan metode *Black Box Testing* untuk memastikan setiap fitur berjalan sesuai kebutuhan, serta pengujian kinerja menggunakan *Chrome DevTools*, *React DevTools*, *Docker Desktop*, dan *Lighthouse* untuk mengevaluasi *page load time*, penggunaan sumber daya sistem (CPU dan memori), kompatibilitas perangkat, serta latensi komunikasi *WebSocket*. Pemilihan lokasi ini didasarkan pada ketersediaan fasilitas pengembangan dan pengujian yang memadai sehingga seluruh proses pengujian dapat dilaksanakan secara terstruktur dan menghasilkan data yang valid.

C. Subjek dan Objek Penelitian

Subjek penelitian ini adalah pengguna sistem yang terdiri dari admin, teknisi, atau operator sistem audio masjid multi-zona yang bertanggung jawab terhadap pemantauan kondisi zona audio. Subjek penelitian menjadi sumber dalam proses identifikasi kebutuhan fungsional *dashboard*, penyusunan skenario pemantauan, serta penentuan informasi yang ditampilkan pada *dashboard web*.

Objek penelitian ini adalah prototipe *dashboard web* untuk *monitoring* sistem audio masjid multi-zona yang dikembangkan menggunakan teknologi *web* dan terhubung dengan layanan *backend* melalui *REST API* dan *WebSocket*. *Dashboard* mencakup tampilan utama status zona audio, halaman detail zona, visualisasi data historis, serta fitur pemantauan kondisi sistem secara *real-time*.

D. Alat dan Bahan

Alat dan bahan yang digunakan dalam penelitian ini terdiri dari perangkat keras dan perangkat lunak yang mendukung proses pengembangan, implementasi, dan pengujian sistem *dashboard web*. Berikut merupakan alat dan bahan yang digunakan dalam penelitian ini:

Tabel 3. 1 Alat yang Digunakan dalam Penelitian

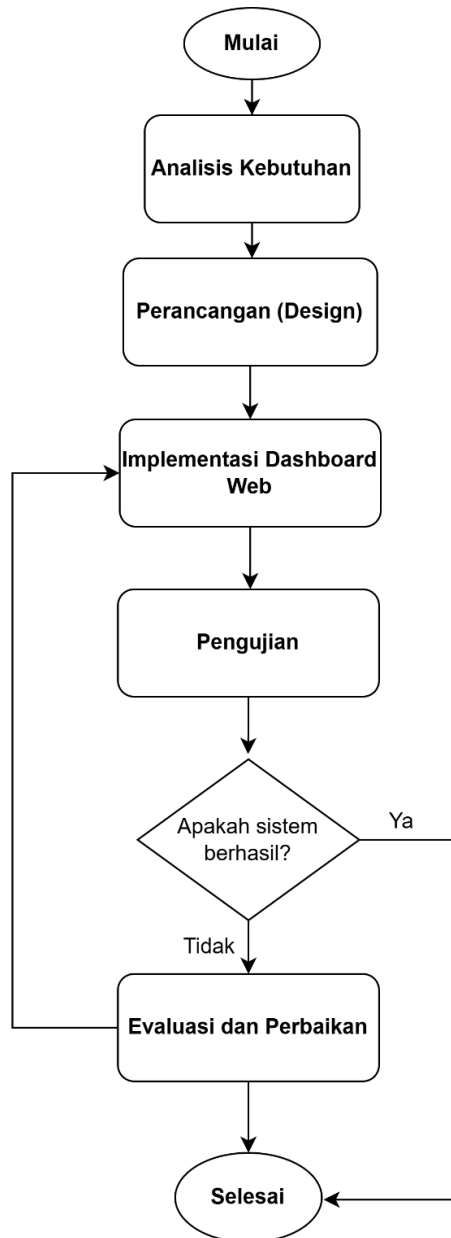
No	Alat	Keterangan
1	Komputer/Laptop	Untuk pengembangan dan pengujian dashboard web.
2	Peramban web <i>modern</i>	Menjalankan dan menguji tampilan dashboard Chrome, Firefox dan Edge
3	Visual Studio Code	Menulis dan mengelola kode program <i>frontend</i> .
4	Koneksi internet	Mengunduh pustaka, dokumentasi, dan melakukan pengujian berbasis <i>web</i> .

Tabel 3. 2 Bahan yang Digunakan dalam Penelitian

No	Bahan	Keterangan
1	React 18.2.0 dan TypeScript v5.3.3	Digunakan untuk membangun antarmuka <i>dashboard web</i> .
2	Python v3.11.2 dan framework FastAPI v0.104.1	Menghubungkan <i>dashboard</i> dengan <i>backend</i> secara waktu nyata.
3	Socket.IO v4.7.2/ Socket.io Client v4.8.3	Digunakan sebagai jalur komunikasi data <i>real-time</i> antara <i>backend</i> dan <i>dashboard</i> .
4	Server backend PostgreSQL v15.15	Digunakan untuk penyimpanan data konfigurasi dan historis sistem.
5	Chart.js v4.5.1 dan Recharts v2.10.3	Digunakan untuk visualisasi data historis dalam bentuk grafik.
6	Data simulasi zona audio	Digunakan sebagai data pengujian kondisi zona audio pada <i>dashboard</i> .

E. Prosedur Penelitian

Prosedur penelitian disusun ke dalam beberapa tahap utama yang saling berkaitan, mengacu pada alur penelitian.



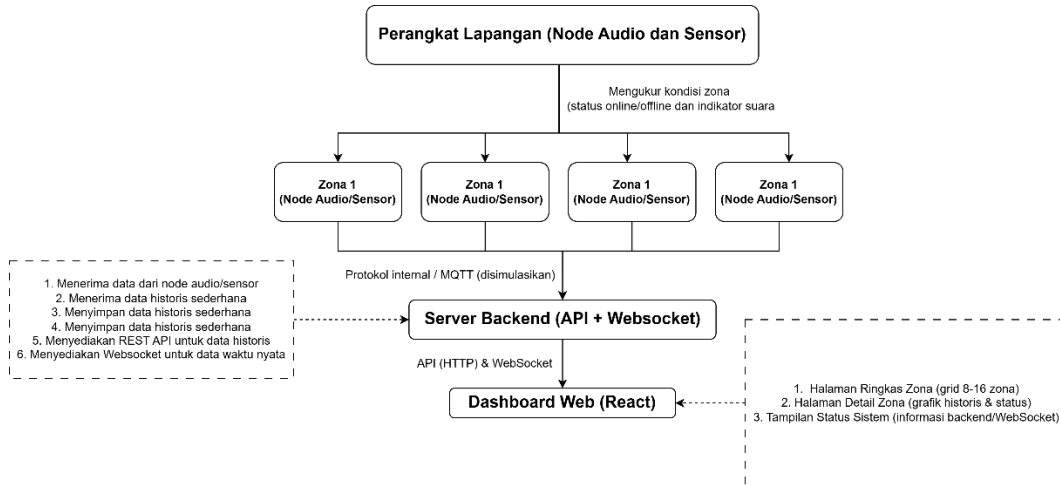
Gambar 3. 1 Diagram Alur Metodologi Penelitian

1. Tahap Analisis Kebutuhan

- a. Mengidentifikasi kebutuhan informasi dan fungsi *dashboard* dari perspektif pengelola sistem audio masjid melalui observasi, wawancara, atau diskusi terbatas dengan narasumber terkait. Dokumentasi diskusi bersama narasumber terkait pada Lampiran 1.
- b. Menyusun kebutuhan fungsional dan *non-fungsional* sistem yang relevan dengan konteks *monitoring* audio masjid multi-zona. Kebutuhan fungsional mencakup fitur utama dashboard, sedangkan kebutuhan *non-fungsional* meliputi aspek kecepatan respons, stabilitas sistem, dan kemampuan komunikasi data secara *real-time* yang relevan dengan konteks pemantauan zona audio.

2. Tahap Perancangan (Desain)

- a. Menyusun rancangan arsitektur sederhana yang menggambarkan hubungan antara *dashboard web*, layanan *backend*, dan jalur komunikasi waktu nyata.

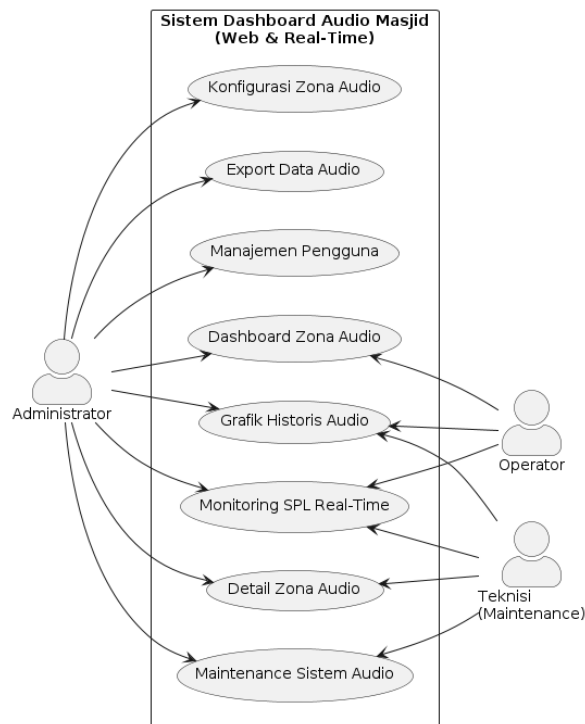


Gambar 3. 2 Arsitektur Sederhana Dashboard Web Sistem Audio Masjid

Gambar 3.2 menunjukkan arsitektur sistem *dashboard monitoring* audio masjid berbasis *Cyber-Physical System (CPS)* yang dirancang untuk memantau kondisi beberapa zona audio secara terpusat dan *real-time*. Arsitektur sistem terdiri dari tiga komponen utama, yaitu perangkat lapangan (node zona audio), server *backend* sebagai pusat pengolahan data, serta *dashboard web* sebagai antarmuka pengguna untuk melakukan proses visualisasi dan *monitoring* informasi sistem.

1) Use Case Diagram

Berdasarkan alur prosedur penelitian yang telah dijelaskan melalui diagram alur, selanjutnya dilakukan pemodelan sistem menggunakan *use case diagram*. *Use case diagram* digunakan untuk menggambarkan interaksi antara pengguna dan sistem *dashboard* audio masjid, serta menjelaskan fungsi-fungsi utama yang dapat diakses oleh masing-masing aktor sesuai dengan perannya.

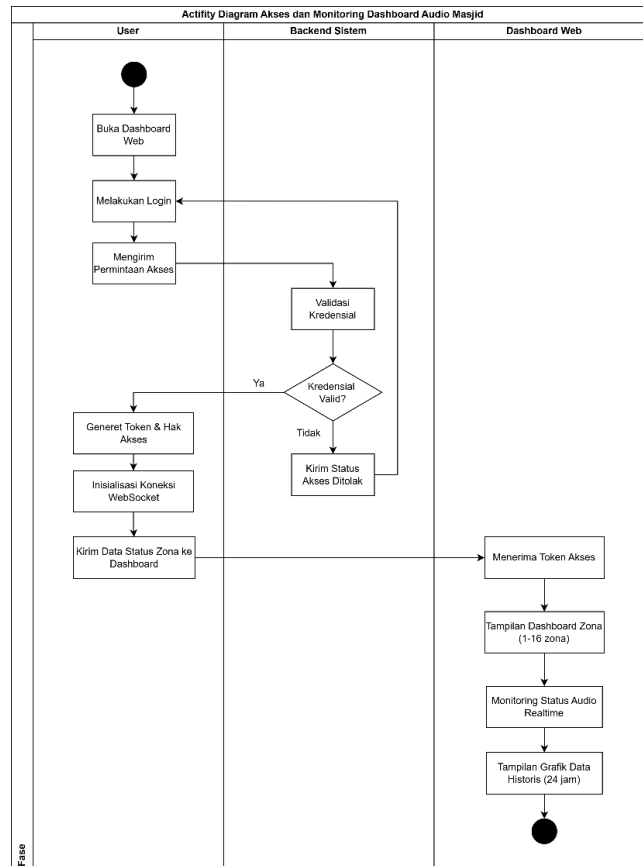


Gambar 3. 3 Use Case Diagram

Gambar 3.3 menunjukkan *Use Case Diagram* pada penelitian yang menggambarkan interaksi antara pengguna dan sistem *dashboard* audio masjid berbasis *web* dan *real-time*. Sistem melibatkan tiga aktor utama, yaitu Admin, Operator, dan Teknisi, yang masing-masing memiliki peran dan hak akses berbeda sesuai kebutuhan operasional. Diagram ini digunakan untuk memperjelas batasan fungsi sistem serta memastikan setiap aktor hanya dapat mengakses fitur yang relevan dengan tugasnya.

Admin memiliki akses penuh terhadap seluruh fitur sistem, meliputi pemantauan zona audio, *monitoring SPL real-time*, serta ekspor data. Operator berfokus pada kegiatan pemantauan kondisi audio melalui dashboard, data *real-time*, dan grafik historis tanpa akses ke pengaturan sistem. Sementara itu, Teknisi memiliki akses ke fitur *monitoring* dan detail zona untuk keperluan pemeliharaan teknis. Pembagian peran ini mendukung pengelolaan sistem audio masjid yang terstruktur, aman, dan efektif dalam mendukung *monitoring* performa audio secara *real-time*.

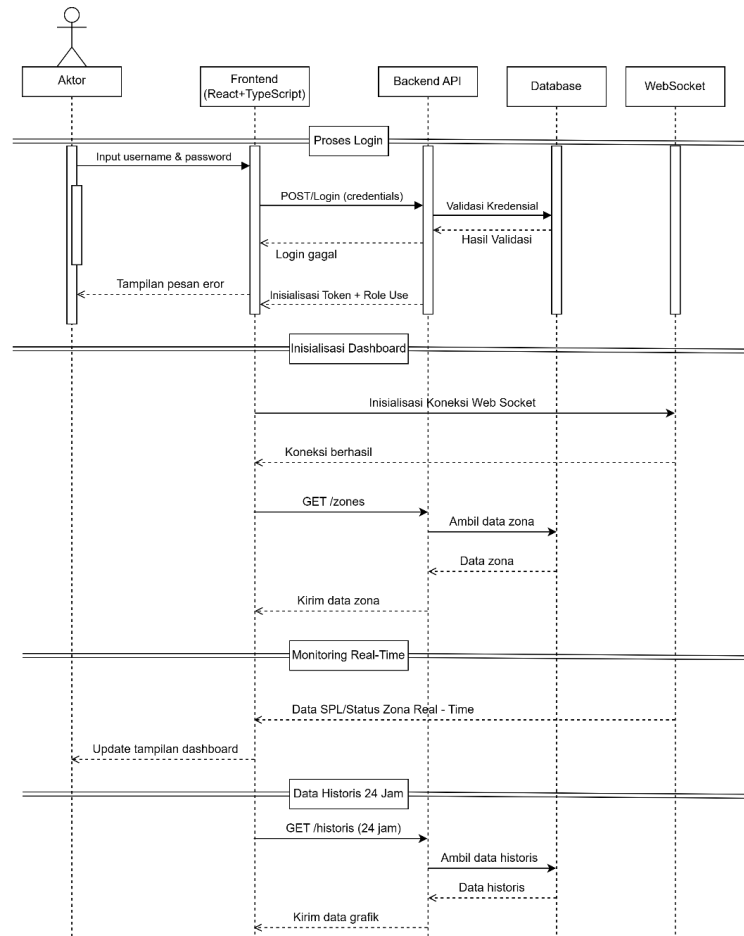
2) Activity Diagram Diagram Akses dan *Monitoring* Audio Masjid



Gambar 3. 4 *Activity Diagram* Akses dan *Monitoring Dashboard* Audio Masjid

Gambar 3.4 menunjukkan *activity diagram* proses autentikasi dan *monitoring dashboard* audio masjid multi-zona. Sistem melakukan validasi kredensial pengguna sebelum memberikan hak akses. Setelah autentikasi berhasil, *backend* menginisialisasi koneksi *WebSocket* untuk mendukung komunikasi data waktu nyata. *Dashboard* kemudian menampilkan status zona audio dan memperbarui informasi secara *real-time* serta menyediakan data historis dalam rentang 24 jam terakhir.

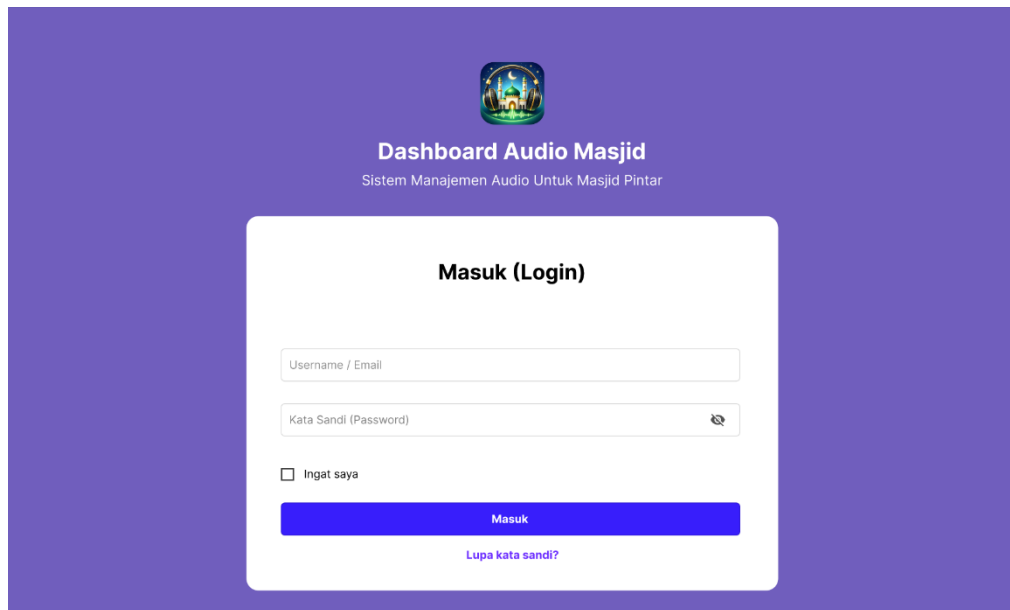
Untuk memperjelas alur interaksi antar komponen sistem setelah proses aktivitas yang telah dijelaskan, diperlukan representasi yang menunjukkan pertukaran pesan secara berurutan antar aktor dan modul sistem. Oleh karena itu, *Sequence Diagram* berikut disusun untuk menggambarkan proses komunikasi antara pengguna, *frontend*, *backend*, *database*, dan layanan *WebSocket* dalam mendukung autentikasi serta *monitoring dashboard* audio masjid multi-zona.



Gambar 3. 5 *Sequence Diagram Login & Monitoring Dashboard Audio*

Gambar 3.5 menunjukkan *Sequence Diagram* yang menggambarkan alur interaksi utama dalam sistem *dashboard* audio masjid, dimulai dari proses autentikasi pengguna hingga *monitoring* zona audio secara *real-time*. Setelah login berhasil dan token diterbitkan oleh *backend*, *frontend* menginisialisasi koneksi *WebSocket* untuk menerima pembaruan data zona secara waktu nyata. Selain itu, sistem juga mengambil data historis 24 jam terakhir melalui *REST API* untuk ditampilkan dalam bentuk grafik. Diagram ini memperlihatkan integrasi antara *frontend*, *backend*, *database*, dan layanan *WebSocket* dalam mendukung *monitoring* multi-zona secara terstruktur dan *responsif*.

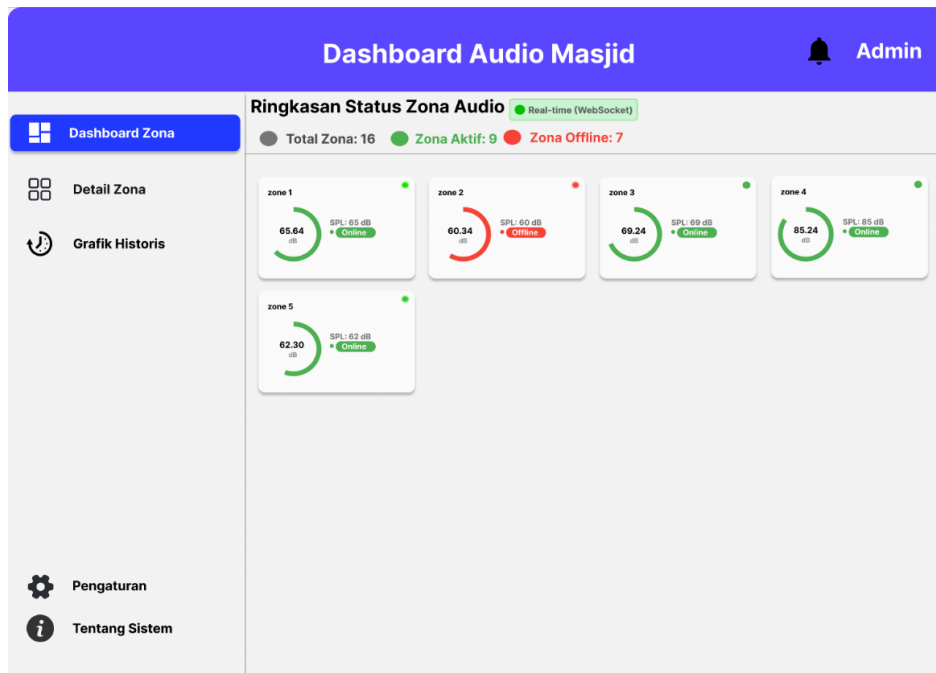
- b. Membuat rancangan antarmuka (*wireframe*) untuk halaman-halaman utama, seperti halaman login, halaman ringkasan zona, halaman detail zona, dan tampilan grafik historis.



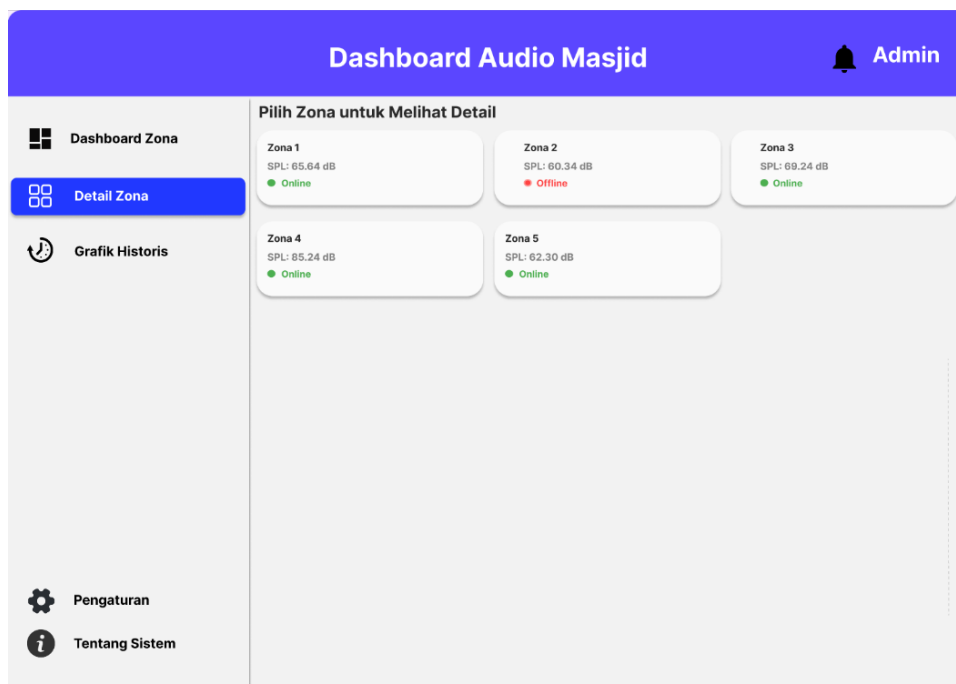
Gambar 3. 6 Halaman Login

Rancangan antarmuka *dashboard web* mencakup empat halaman utama, seperti ditunjukkan pada Gambar 3.6. Halaman pertama adalah halaman login (Gambar 3.6) yang memastikan admin, operator dan teknisi terotorisasi yang dapat mengakses sistem. Setelah login, pengguna masuk ke halaman ringkasan zona (Gambar 3.7) yang menampilkan status semua 8–16 zona dalam satu pandangan terpusat dengan indikator status berwarna dan nilai *SPL* saat ini.

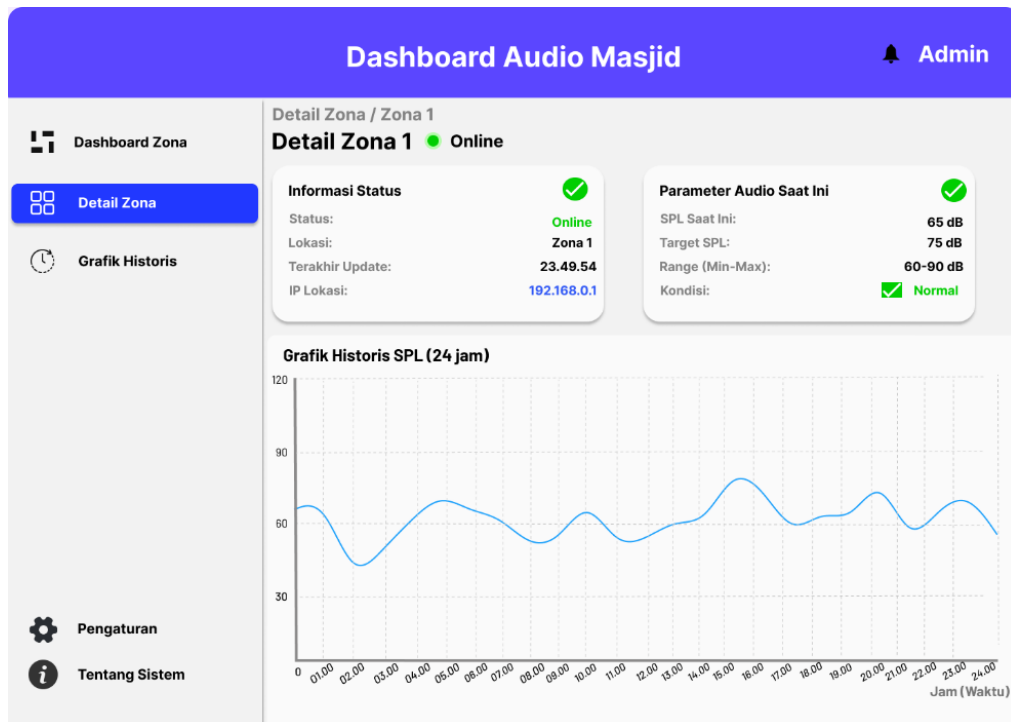
Dari halaman ringkasan, pengguna dapat membuka halaman detail zona (Gambar 3.8) untuk melihat informasi lebih rinci tentang zona tertentu dan (Gambar 3.9) untuk melihat halaman detail dari zona termasuk grafik *SPL* 24 jam. Untuk analisis data jangka menengah dan perbandingan antar zona, tersedia halaman grafik historis (Gambar 3.10) yang dilengkapi dengan filter, visualisasi ganda, dan opsi export data. Keempat halaman ini dirancang dengan mempertimbangkan kebutuhan admin, operator dan teknisi masjid yang memerlukan *monitoring* cepat, analisis detail, dan kemampuan *export* laporan



Gambar 3. 7 Halaman Ringkasan Zona



Gambar 3. 8 Halaman Ringkasan Detail Zona



Gambar 3. 9 Halaman Detail dari Zona

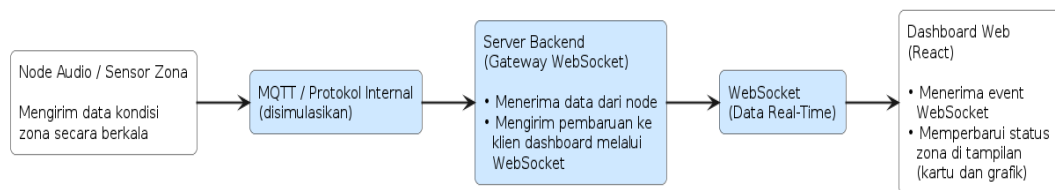


Gambar 3. 10 Halaman Grafik Historis

Tabel 3. 3 Ringkasan halaman utama *Dashboard Web*

No	Halaman	Fungsi Utama	Komponen Kunci
1	Login	Autentikasi pengguna	<i>Form login, field email/password, tombol masuk</i>
2	Ringkasan Zona	Pandangan umum semua zona	<i>Grid zona 4×4, indikator status, SPL saat ini, info ringkas</i>
3	Detail Zona	Analisis zona spesifik	Informasi status, parameter audio, grafik 24 jam
4	Grafik Historis	Analisis data jangka menengah	Filter zona/rentang waktu, grafik ganda, tabel statistik, <i>export</i>

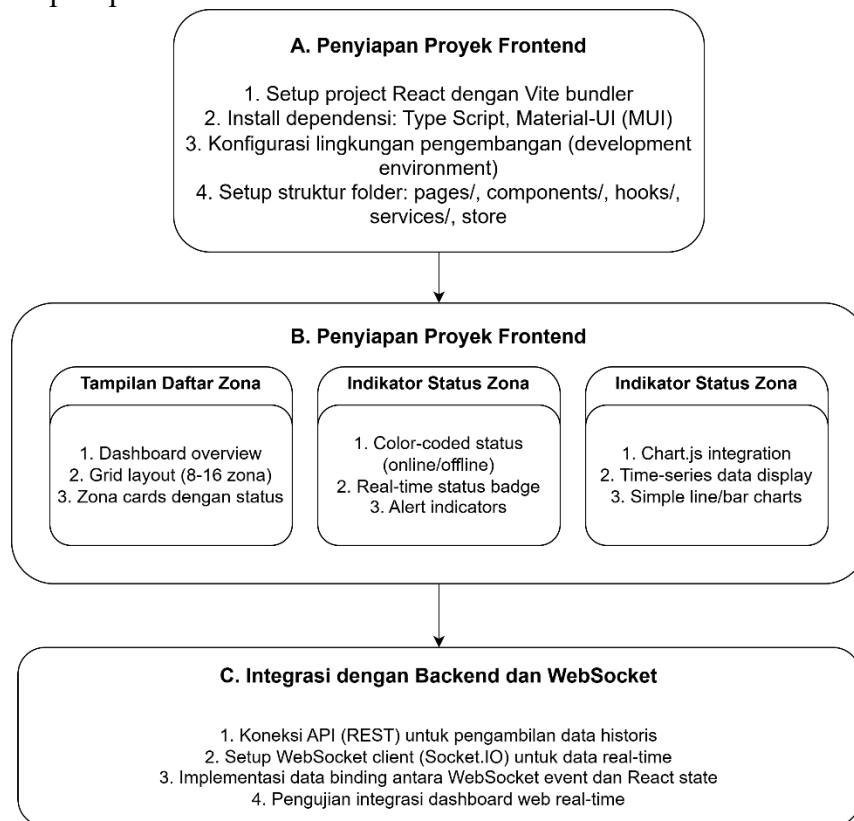
- c. Menentukan struktur data yang digunakan pada dashboard (misalnya data zona, status, dan nilai pengukuran) serta rancangan umum *endpoint API* dan saluran *WebSocket* yang diperlukan.



Gambar 3. 11 Alur Data *WebSocket* pada *Dashboard Web*

Gambar 3.11 menunjukkan alur komunikasi data *real-time* menggunakan *WebSocket* antara *backend* dan *dashboard web*. Data kondisi zona yang dikirim oleh node audio diterima terlebih dahulu oleh *backend*, kemudian didistribusikan kepada *dashboard* melalui koneksi *WebSocket*. Dashboard menerima setiap pembaruan data secara langsung dan memperbarui tampilan status zona serta grafik pemantauan tanpa perlu melakukan proses polling secara berkala. Mekanisme ini memungkinkan informasi kondisi sistem ditampilkan secara *real-time* dengan latensi yang rendah.

3. Tahap Implementasi



Gambar 3. 12 Tahap Implementasi *Dashboard Web*

- a. Menyiapkan proyek *frontend* menggunakan kerangka kerja *React* dan teknologi pendukung yang diperlukan untuk membangun tampilan *dashboard*.
- b. Mengimplementasikan komponen antarmuka utama, termasuk tampilan daftar zona, indikator status zona, dan visualisasi data historis dalam bentuk grafik.
- c. Mengintegrasikan *dashboard* dengan layanan *backend* atau server simulasi melalui *REST API* untuk pengambilan data dan *WebSocket* untuk pembaruan data waktu nyata.

4. Tahap Pengujian

Sebelum pengujian dilakukan, seluruh lingkungan pengujian divalidasi terlebih dahulu untuk memastikan kesiapan perangkat keras, perangkat lunak, basis data, jaringan, akun pengguna, serta data simulasi yang digunakan selama penelitian. Proses validasi mencakup pemeriksaan server *dashboard*, *container frontend* dan *backend*, *PostgreSQL*, perangkat klien, dataset historis *SPL*, serta layanan *REST API* dan *WebSocket*. Dokumentasi lengkap validasi lingkungan pengujian disajikan pada Lampiran 6. Berikut tahap pengujian :

- a. Melakukan pengujian fungsional untuk memastikan fitur utama dashboard berjalan sesuai dengan kebutuhan, seperti tampilan zona, pembaruan status, dan pemuatan grafik.

Pengujian fungsional dilakukan dengan skenario uji terstruktur, seperti ditunjukkan pada (Gambar 3.13) dan (Tabel 3.4). Sepuluh fitur utama *dashboard* diuji, meliputi: login, tampilan daftar zona, indikator status zona, halaman detail zona, pemutakhiran data real-time melalui *WebSocket*, grafik *historis*, filter zona pada grafik, *export* data, dan *responsive design*. Setiap skenario pengujian dirancang dengan langkah-langkah eksplisit dan hasil yang diharapkan, sehingga pengujian dapat dilakukan secara konsisten dan hasil dapat dicatat dengan jelas. Fitur dianggap lolos pengujian jika hasil aktual sesuai dengan hasil yang diharapkan.

Tabel 3. 4 Skenario Pengujian Fungsional *Dashboard*

No	Fitur yang Diuji	Skenario Pengujian	Kriteria Keberhasilan	Hasil	Status
F01	<i>Login</i>	Admin memasukkan <i>password</i> benar	Berhasil masuk <15 detik, token <i>JWT</i> tersimpan	0.704 ms, token OK	Lulus
F02	<i>Login (Negative)</i>	Password salah	<i>Error message</i> muncul, tetap di <i>login page</i>	1.162 ms, <i>Error</i> ditampilkan	Lulus
F03	<i>Dashboard Overview</i>	Tampilan 5 zona dari 16 zona audio	Semua zona terlihat, status akurat	16 zona <i>visible</i> , semua render	Lulus
F04	Indikator Status	Zona online = hijau, offline = merah	Warna konsisten dengan status <i>database</i>	Hijau (#4CAF50) + pulse, Merah (#F44336) Warna sesuai	Lulus
F05	Halaman Detail Zona	Klik zona untuk lihat detail	Halaman terbuka < 15 detik, data lengkap	2.91 detik ambil rata-rata dari semua zona, data OK	Lulus
F06	<i>Real-time Update WebSocket</i>	Status zona berubah di <i>backend</i>	<i>Dashboard update</i> < 3 detik tanpa refresh	0.158 detik	Lulus
F07	Grafik Historis <i>SPL</i>	Tampilan grafik 24 jam (1440 points)	Grafik render < 4 detik, <i>tooltip</i> interaktif	1.667 detik, <i>Recharts tooltip</i> OK	Lulus
F08	Filter Zona Grafik	Pilih zona spesifik	Grafik <i>update</i> sesuai zona dipilih	Data Zona 5 real dari 16 zona	Lulus
F09	<i>Export Data CSV</i>	<i>Ekspor data</i> ke <i>CSV</i>	File terunduh < 5 detik, format valid	0.04 detik, <i>CSV</i> OK	Lulus

F10	<i>Responsive Design</i>	Akses dari desktop, tablet, mobile	<i>Layout adaptif, semua fitur accessible</i>	Tampilan desktop, tablet dan mobile <i>responsive</i>	Lulus
F11	<i>Logout</i>	Klik logout	Sesi berakhir, <i>redirect</i> ke login	<i>Logout</i> berhasil	Lulus

- b. Melakukan pengujian kinerja sederhana melalui waktu muat halaman awal dan respons pembaruan data dari *WebSocket* pada skenario simulasi. Pengujian kinerja dilakukan untuk mengukur performa *dashboard web* dalam kondisi penggunaan normal.

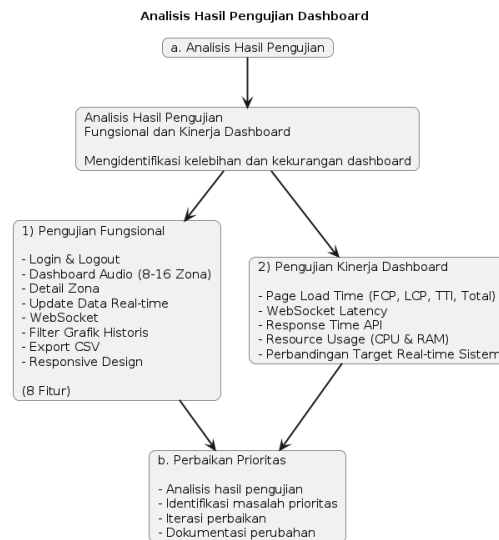
Tabel 3. 5 Skenario Pengujian Kinerja *Dashboard Web* Sistem Audio Masjid

No	Metrik Kinerja	Target	Hasil	Status	Ref.
A. Page Load Time					
P01	<i>Login page (FCP)</i>	< 3.0 detik	1.240 detik	Lulus	MDPI T6
P02	<i>Login page (LCP)</i>	< 3.0 detik	0,172 detik	Lulus	MDPI T6
P03	<i>Login page (TTI)</i>	< 3.0 detik	1.240 detik	Lulus	MDPI T6
P04	<i>Dashboard overview (FCP)</i>	< 3,5 detik	1.032 detik	Lulus	MDPI T6
P05	<i>Dashboard overview (LCP)</i>	< 3.5 detik	0,190 detik	Lulus	MDPI T6
P06	<i>Dashboard overview (TTI)</i>	< 3.5 detik	1.982 detik	Lulus	MDPI T6
P07	<i>Detail zona (Total)</i>	< 3.0 detik	2.258 detik	Lulus	MDPI T6
P08	<i>Grafik 1440 points (Total)</i>	< 4.0 detik	3.856 detik	Lulus	MDPI T6
P09	<i>Rata-rata Page Load</i>	≤ 2.75 detik	3.15 detik	Perlu Optimasi	MDPI T6
B. WebSocket Latency					
P10	<i>Latency minimum</i>	≤ 20 ms	8.2 ms	Lulus	MDPI T8
P11	<i>Latency rata-rata</i>	≤ 50 ms	18.5 ms	Lulus	MDPI T8
P12	<i>Latency maksimum</i>	≤ 100 ms	61.4 ms	Lulus	MDPI T8
P13	<i>P95 latency</i>	≤ 60 ms	32.1 ms	Lulus	MDPI T8
P14	<i>P99 latency</i>	≤ 80 ms	41.3 ms	Lulus	MDPI T8
P15	<i>Packet loss</i>	≤ 0.1%	0 %	Lulus	MDPI T8

P16	<i>Uptime koneksi (1 jam test)</i>	$\geq 99.5\%$	100%	Lulus	MDPI T8
C. Resource Usage					
P17	<i>Memori (Idle 5 menit)</i>	< 150 MB	93,9 MB	Lulus	MDPI T7
P18	<i>Memori (Live update 10 menit)</i>	< 180 MB	95,5 MB	Lulus	MDPI T7
P19	<i>Memori (Grafik + update)</i>	< 210 MB	120,4 MB	Lulus	MDPI T7
P20	<i>Memori (Peak)</i>	< 250 MB	185,2 MB	Lulus	MDPI T7
P21	<i>CPU (Idle)</i>	< 5%	1,2%	Lulus	MDPI T7
P22	<i>CPU (Live update)</i>	< 10%	2,8%	Lulus	MDPI T7
P23	<i>CPU (Grafik + update)</i>	< 18%	35,1%	Tidak Lulus	MDPI T7
P24	<i>CPU (Peak)</i>	< 30%	28,5%	Lulus	MDPI T7
D. Operasi Khusus					
P25	Waktu filter grafik	< 2.0 detik	1.670 detik	Lulus	Skripsi T3.4
P26	Waktu <i>export CSV</i>	< 5.0 detik	0.040 detik	Lulus	Skripsi T3.4
P27	Waktu render zona <i>card (update)</i>	< 500 ms	158 ms	Lulus	MDPI T7
P28	Waktu respons <i>API (/zones)</i>	< 100 ms	0.357 ms	Perlu Optimasi	MDPI T2

Seperti ditunjukkan pada Tabel 3.5, 28 metrik kinerja utama diukur, meliputi: waktu muat halaman awal, latensi *WebSocket* untuk pembaruan data *real-time*, waktu render zona *card*, waktu rendering grafik historis, penggunaan memori pada kondisi *idle* dan *live update*, penggunaan *CPU* pada kondisi *idle* dan *peak*, waktu eksekusi filter dan *export*, serta waktu respons *API*. Pengujian dilakukan pada kondisi jaringan yang wajar (4G atau WiFi ~10 Mbps) dengan *hardware* standar (laptop mid-range), sehingga hasil pengujian mencerminkan performa pada kondisi penggunaan nyata. Alat pengukuran yang digunakan adalah *Chrome DevTools* bawaan peramban, *React DevTools* untuk profiling komponen, dan *Task Manager* untuk *monitoring resource*.

5. Tahap Evaluasi dan Perbaikan



Gambar 3. 13 Tahap Evaluasi dan Perbaikan *Dashboard Web*

- a. Menganalisis hasil pengujian fungsional dan kinerja untuk mengidentifikasi kelebihan dan kekurangan *dashboard*.
- b. Melakukan perbaikan pada komponen *dashboard* yang masih belum memenuhi target pengujian, seperti optimasi waktu muat halaman, peningkatan performa rendering grafik, atau stabilitas pembaruan data *real-time* melalui *WebSocket*.

Hasil pengujian fungsional akan dianalisis dengan membandingkan hasil eksekusi setiap fitur terhadap spesifikasi yang telah ditetapkan. Pengujian kinerja akan dianalisis dengan menghitung rata-rata metrik (waktu muat, latensi, penggunaan memori) dan membandingkannya dengan target yang realistis untuk prototipe S1.

F. Jadwal Penelitian

Penelitian dan pengembangan *dashboard web* untuk sistem *monitoring* zona sistem audio masjid direncanakan berlangsung selama enam bulan, yaitu dari Januari hingga Juni 2026. Jadwal penelitian disusun dengan mempertimbangkan durasi setiap tahap pengembangan, sehingga pekerjaan dapat berjalan secara terstruktur dan tepat waktu.

Tabel 3. 6 Jadwal Penelitian

No	Kegiatan	Bulan 1 (Jan)	Bulan 2 (Feb)	Bulan 3 (Mar)	Bulan 4 (Apr)	Bulan 5 (Mei)	Bulan 6 (Jun)
1	Penyusunan Rencana Penelitian	✓					
2	Studi Literatur dan Pengumpulan Referensi	✓					
3	Analisis Kebutuhan dan Wawancara dengan narasumber terkait	✓					
4	Perancangan Arsitektur Sistem		✓				
5	Perancangan <i>Wireframe</i> dan <i>UI/UX Dashboard</i>		✓				
6	Setup Proyek dan Implementasi <i>Foundation</i> (Sprint 1)		✓				
7	Implementasi Komponen <i>UI</i> (Sprint 2)			✓			
8	Integrasi <i>REST API</i> dan <i>WebSocket Real-time</i> (Sprint 3)			✓			
9	Implementasi Fitur Lanjutan (Sprint 4)			✓	✓		
10	Penyusunan tahap Pengujian <i>Dashboard</i> dan Refrensi terkait pengujian				✓		
10	Pengujian Fungsional <i>Dashboard</i>				✓		
11	Pengujian Kinerja <i>Dashboard</i> (Performance Testing)				✓		
12	Pengujian konektifitas <i>Real-time WebSocket</i>				✓	✓	
13	Analisis Hasil Pengujian				✓	✓	
14	Perbaikan dan Optimasi <i>Dashboard</i>					✓	✓
15	Dokumentasi Final dan Persiapan Laporan						✓
16	Penyusunan Laporan Skripsi (BAB I–V)						✓
17	Review, Revisi, dan Persiapan Sidang						✓

BAB IV HASIL DAN PEMBAHASAN

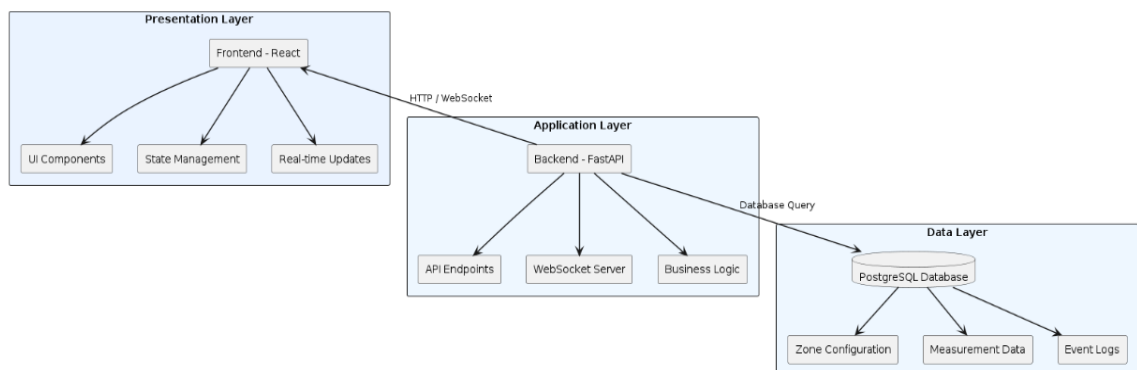
A. Hasil Implementasi Dashboard Web

1. Deskripsi Implementasi Sistem

Dashboard web yang dikembangkan merupakan aplikasi berbasis *web* dengan arsitektur *client-server* yang berfungsi untuk *monitoring* sistem audio masjid multi-zona secara *real-time*. Sistem ini dibangun untuk menampilkan informasi kondisi zona audio melalui integrasi layanan *backend* menggunakan *FastAPI* dan *WebSocket*. *Dashboard* menyajikan data status zona, nilai tingkat tekanan suara *SPL*, serta informasi historis dalam bentuk visualisasi grafik untuk mendukung proses pemantauan sistem audio masjid secara terpusat.

a. Arsitektur Sistem *Dashboard*

Dashboard web dibangun dengan arsitektur tiga lapis (*three-tier architecture*) seperti ditunjukkan pada Gambar 4.1:



Gambar 4. 1 Arsitektur Sistem *Dashboard Web* Tiga Lapis

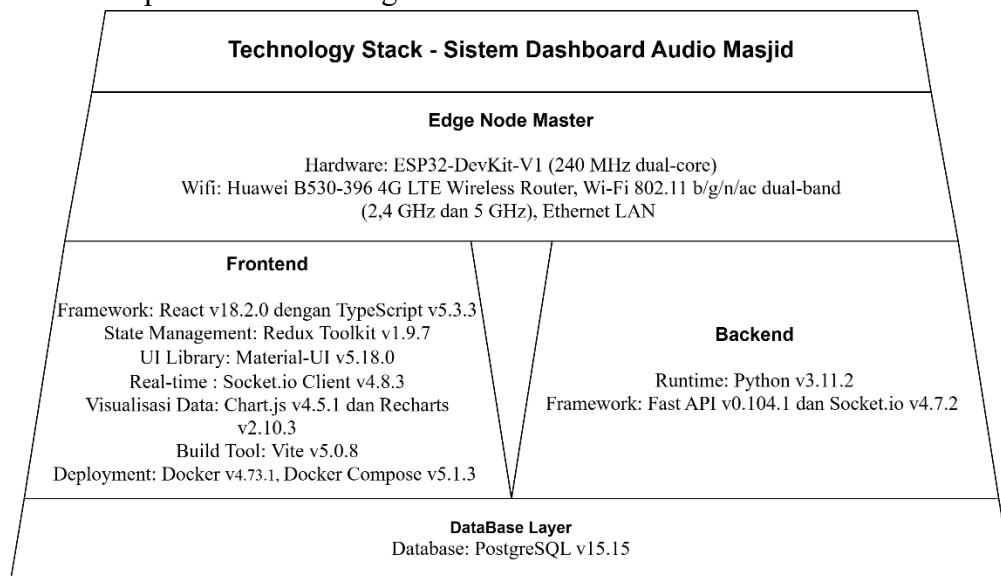
Arsitektur sistem *dashboard* ini dibangun dalam tiga lapis utama, yaitu *presentation layer*, *application layer*, dan *data layer*, agar pemrosesan tampilan, logika aplikasi, dan penyimpanan data terpisah dengan jelas serta mudah dikembangkan.

Pada *presentation layer* (*Frontend – React*), *dashboard* menyediakan antarmuka *web* responsif untuk memonitor 8–16 zona audio secara sekaligus, menampilkan status zona, nilai *SPL*, serta parameter pendukung lain dalam bentuk kartu dan grafik. Pengelolaan state dilakukan dengan *Redux Toolkit* sehingga perubahan data dari server dapat terdistribusi konsisten ke seluruh komponen, sementara *Socket.io* dimanfaatkan untuk menerima pembaruan *real-time* dan *Chart.js/Recharts* digunakan untuk memvisualisasikan data historis *SPL* dan metrik lain secara interaktif.

Pada *application layer (Backend – FastAPI)*, server *FastAPI* menyediakan *REST API* untuk menyediakan data *SPL*, detail zona, dan data historis dan juga mengelola server *WebSocket* berbasis *Socket.io* untuk mendorong (*push*) *update real-time* ke *klien* ketika ada perubahan data di sisi perangkat atau basis data. Lapis ini juga bertugas memproses logika utama sistem, mulai dari memvalidasi data masuk, memformat struktur data historis, membangkitkan notifikasi status koneksi layanan *REST API* dan *WebSocket*, hingga memvalidasi sesi akses (*login*) pengguna.

Pada data layer (*PostgreSQL Database*), seluruh konfigurasi zona audio, data pengukuran *SPL*, dan parameter lingkungan lain dari tiap zona disimpan dalam bentuk penyimpanan struktur *time-series*. Pola penyimpanan ini dioptimasi agar sistem dapat melakukan perekaman data secara berkesinambungan dan merespons kueri rentang waktu dengan cepat, sehingga *dashboard* mampu menyediakan *monitoring* historis grafik *SPL* secara terpadu untuk analisis tren kebisingan di lingkungan masjid.

b. Spesifikasi Teknologi



Gambar 4. 2 *Technology Stack* - Sistem Dashboard Audio Masjid

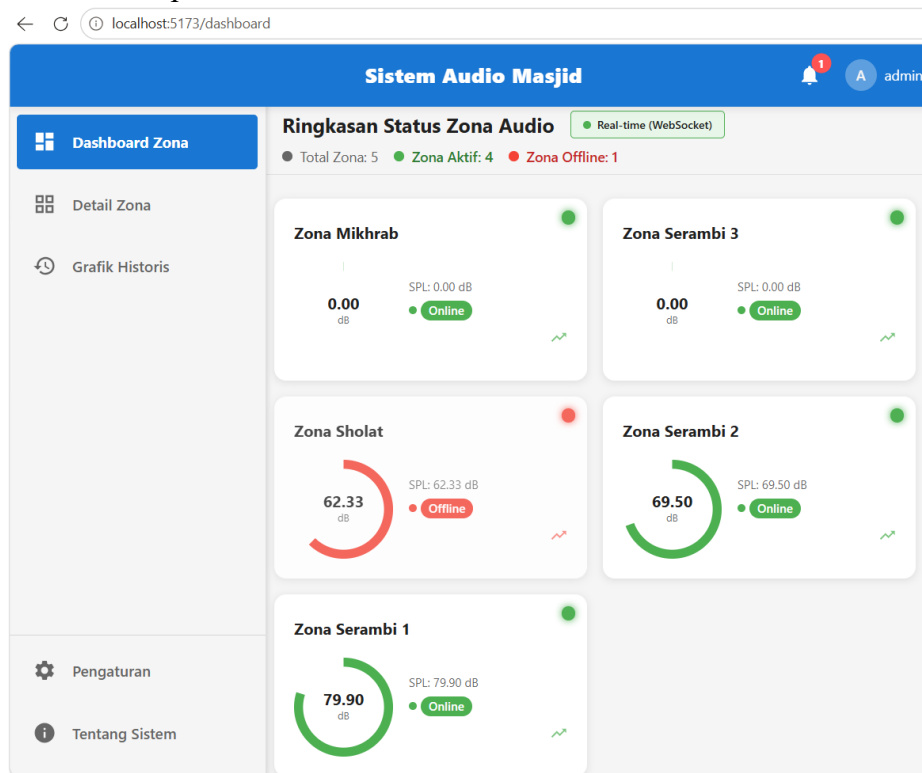
Gambar 4.2 menunjukkan Stack teknologi yang digunakan pada pengembangan *dashboard* dibagi menjadi tiga bagian, yaitu *frontend*, *backend*, dan integrasi perangkat keras. Pada sisi *frontend*, sistem menggunakan *React 18.2.0* dengan *TypeScript* yang dipadukan dengan *Redux Toolkit* untuk manajemen *state*, *Material-UI v5.18.0* sebagai pustaka komponen antarmuka, *Socket.io Client v4.8.3* untuk komunikasi *real-time*, serta *Chart.js v4.5.1* dan *Recharts v2.10.3* untuk visualisasi data, dengan proses build ditangani oleh *Vite versi 5.0.8* yang sudah mendukung *SSR-ready*. Pada sisi *backend*, dashboard berjalan di atas *Python 3.11.2* dengan *framework FastAPI 0.104.1*, menggunakan *Socket.io v4.7.5* sebagai protokol *real-time*, *PostgreSQL 15.15* yang dioptimasi untuk data *time-series*

sebagai basis data utama, dan *di-deploy* menggunakan *Docker v4.73.1* serta *Docker Compose v5.1.3* untuk memudahkan pengelolaan layanan. Untuk integrasi hardware, sistem terhubung ke *Edge Node Master* berbasis *ESP32-DevKit-V1 (240 MHz dual-core)* yang berkomunikasi melalui *Huawei B530-396 4G LTE Wireless Router* dan *Wi-Fi 802.11 b/g/n/ac dual-band (2,4 GHz dan 5 GHz)*, *Ethernet LAN*.

Tabel 4. 1 Spesifikasi Teknologi

Lapisan	Komponen	Spesifikasi
Frontend	Framework	React v18.2.0 dengan TypeScript v5.3.3
	State Management	Redux Toolkit v1.9.7
	UI Component Library	Material-UI v5.18.0
	Real-time Communication	Socket.io Client v4.8.3
	Real-time Protocol	Socket.io v4.7.2
	Visualisasi Data	Chart.js v4.5.1 dan Recharts v2.10.3
	Build Tool	Vite v5.0.8
Backend	Deployment	Docker v4.73.1, Docker Compose v5.1.3
	Runtime	Python v3.11.2
	Framework	FastAPI v0.104.1 dan Socket.io v4.7.2
	Database	PostgreSQL v15.15 (time-series optimized)
Hardware Integration	Edge Node Master	ESP32-DevKit-V1 (240 MHz dual-core)
	Komunikasi Jaringan	Huawei B530-396 4G LTE Wireless Router, Wi-Fi 802.11 b/g/n/ac dual-band (2,4 GHz dan 5 GHz), Ethernet LAN

c. Mockup Interface *Dashboard*

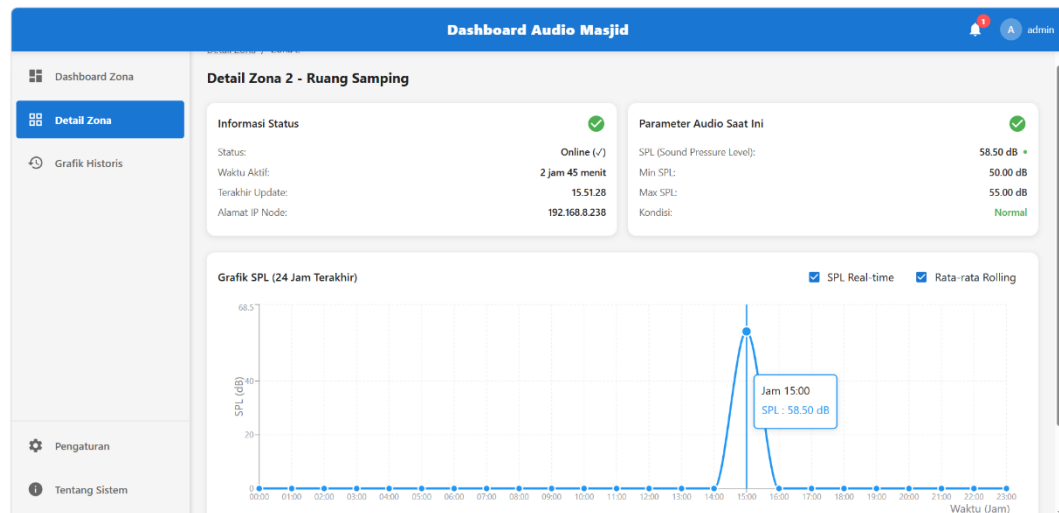


Gambar 4. 3 Dashboard Overview

Gambar 4.3 menunjukkan rancangan antarmuka (mockup) *dashboard* overview yang dirancang untuk mendukung pemantauan hingga 8–16 zona audio dalam satu tampilan terpusat. Setiap zona direpresentasikan dalam bentuk kartu yang menampilkan informasi nama zona, status koneksi, nilai SPL, dan indikator kekuatan sinyal.

Pada tahap implementasi dan pengujian, sistem menggunakan 5 zona audio sebagai sampel untuk memvalidasi fungsi *monitoring* dan komunikasi data real-time. Meskipun demikian, struktur antarmuka yang dirancang tetap mendukung pengembangan hingga 16 zona audio sesuai kebutuhan sistem yang telah ditetapkan. Setiap zona card menampilkan:

- Nama zona
- Status online/offline (indikator warna hijau/merah)
- *SPL* saat ini (dB)
- Status *real-time Websocket*



Gambar 4. 4 Mockup Halaman Detail Zona dengan Grafik *SPL* 24 Jam

Halaman detail zona menampilkan informasi detail termasuk *metrics* utama dan grafik *SPL* 24 jam untuk analisis trend.

2. Hasil Pengujian Fungsional Dashboard

a. Pengujian Fitur Utama

Pengujian fungsional pada penelitian ini menggunakan metode *Black Box Testing* untuk memverifikasi bahwa setiap fitur dashboard menghasilkan keluaran yang sesuai dengan kebutuhan sistem. Pengujian dilakukan pada 8 fitur utama dashboard untuk memastikan setiap komponen berfungsi sesuai spesifikasi.

Tabel 4. 2 Hasil Pengujian Fitur Dashboard

No	Fitur	Skenario Pengujian	Hasil	Status
1	<i>Autentikasi</i>	Login dengan <i>username</i> dan <i>password</i>	Berhasil masuk, sesi aktif	✓ Lulus
2	<i>Dashboard Overview</i>	Menampilkan status 8 -16 zona	Semua zona terlihat status akurat	✓ Lulus
3	<i>Status Indikator</i>	Zona online berwarna hijau, offline merah	Status zona berubah sesuai kondisi perangkat	✓ Lulus
4	<i>Detail Zona</i>	Klik zona menampilkan halaman detail <i>SPL</i> , status	Informasi <i>SPL</i> dan status zona berhasil ditampilkan	✓ Lulus
5	<i>Grafik Historis</i>	Menampilkan grafik <i>SPL</i> 24 jam (1440 data point)	Grafik responsif, <i>zoom</i> berfungsi	✓ Lulus
6	<i>Filter Data</i>	Memilih zona dari <i>dropdown</i>	Grafik terupdate instant	✓ Lulus
7	<i>Export Data</i>	Mengekspor ke <i>CSV</i>	<i>File CSV valid</i> , terbuka di <i>Excel</i>	✓ Lulus
8	<i>Logout</i>	Keluar dari sistem	Sesi berakhir, kembali ke login	✓ Lulus

8/8 fitur lulus pengujian (100%). Dashboard siap untuk digunakan.

b. Pengujian Real-Time Update Dashboard

Pengujian *real-time* update dilakukan untuk memverifikasi kemampuan *dashboard* dalam menerima perubahan status zona melalui *WebSocket*. Pengujian menggunakan 5 zona audio sebagai sampel implementasi, meskipun *dashboard* dirancang untuk mendukung *monitoring* hingga 16 zona audio. Skenario pengujian dilakukan dengan mensimulasikan perubahan status zona dan mengamati respons *dashboard* terhadap data yang diterima secara *real-time*.

Tabel 4. 3 Hasil Pengujian Real-Time Update Dashboard

No	Zona yang diuji	Perubahan (Online/Offline)	Latency UI Update (ms)	Jumlah Pesan WS Diterima	Packet Loss (Ya/Tidak)	Hasil (Lulus/Gagal)	Catatan
1	Zona 1	Online → Offline	248 ms	1	Tidak	LULUS	Latensi meningkat karena <i>dashboard</i> melakukan pembaruan status zona sekaligus rendering visualisasi SPL sebesar 79,20 dB.
2	Zona 2	Online → Offline	221 ms	1	Tidak	LULUS	Latensi meningkat karena <i>dashboard</i> melakukan pembaruan status zona sekaligus rendering visualisasi SPL sebesar 69,50 dB.
3	Zona 3	Online → Offline	252 ms	1	Tidak	LULUS	Latensi tertinggi terjadi saat <i>dashboard</i> melakukan sinkronisasi status zona dan pembaruan visualisasi SPL sebesar 62,33 dB secara bersamaan.
4	Zona 4	Online → Offline	26 ms	1	Tidak	LULUS	Sangat optimal. Nilai latensi sukses mendekati standar target pengujian (P99) karena Beban teringan grafik kosong (0 dB), murni menampilkan kecepatan jaringan WiFi.
5	Zona 5	Online → Offline	44 ms	1	Tidak	LULUS	Sangat optimal. Nilai latensi sukses mendekati standar target pengujian

							(P99) karena Beban teringan grafik kosong (0 dB), murni menampilkan kecepatan jaringan WiFi.
--	--	--	--	--	--	--	--

Berdasarkan hasil pengujian pada Tabel 4.3, seluruh zona sampel berhasil menerima pembaruan status secara *real-time* tanpa kehilangan paket data (*packet loss 0%*). Jumlah pesan *WebSocket* yang diterima pada setiap pengujian konsisten sebanyak satu pesan, yang menunjukkan bahwa setiap perubahan status hanya menghasilkan satu *event* pembaruan sehingga komunikasi data berlangsung secara efisien. Nilai *latency UI update* berada pada rentang 26–252 ms, dengan Zona 4 dan Zona 5 mencatat latensi terendah karena hanya memerlukan pembaruan status sederhana tanpa beban visualisasi *SPL* yang signifikan, sedangkan Zona 1, Zona 2, dan terutama Zona 3 menunjukkan latensi lebih tinggi karena komponen *React* harus melakukan pembaruan *state*, *rendering* ulang visualisasi *SPL*, dan sinkronisasi tampilan setelah menerima data dari *WebSocket*. Meskipun terdapat variasi waktu pembaruan antarmuka, seluruh perubahan status berhasil ditampilkan dengan benar sehingga mekanisme *monitoring real-time* berbasis *React*, *Redux Toolkit*, dan *WebSocket* dapat dinyatakan berjalan dengan baik dan stabil.

Hasil ini sejalan dengan penelitian Chodorek dan Chodorek (2025) yang mengevaluasi komunikasi data *real-time* menggunakan *WebSocket* melalui pengukuran *end-to-end latency* dan keandalan pengiriman data. Hasil evaluasi pada penelitian tersebut menunjukkan bahwa *WebSocket* mampu mempertahankan komunikasi dengan latensi rendah sehingga sesuai digunakan pada aplikasi yang memerlukan pembaruan data secara *real-time*. Oleh karena itu, penelitian tersebut dipilih sebagai referensi pendukung karena menggunakan parameter evaluasi yang sama dengan penelitian ini, yaitu *latency* dan keberhasilan pengiriman data. Meskipun penelitian tersebut tidak menetapkan target numerik tertentu, hasil evaluasinya memperkuat bahwa implementasi *WebSocket* pada penelitian ini telah mampu mendukung *monitoring* sistem audio multi-zona secara cepat dan stabil. Dokumentasi lengkap pengujian *real-time update* disajikan pada Lampiran 2.

c. Pengujian *Responsivitas* dan *Kompatibilitas*

Dashboard diuji pada berbagai ukuran layar dan *browser* untuk memastikan *responsivitas* optimal.

Tabel 4. 4 Hasil Pengujian *Kompatibilitas Device*

Device	Resolusi	Browser	Status Layout	Fitur	Status
Desktop	1920×1080	Chrome v120	✓ Optimal	✓ Semua	✓ Lulus
Laptop	1366×768	Firefox v121	✓ Baik	✓ Semua	✓ Lulus

Tablet	1024×768	Safari v17	✓ Baik	✓ Semua	✓ Lulus
--------	----------	------------	--------	---------	------------

Kesimpulan: Berdasarkan hasil pengujian *kompatibilitas* perangkat, *dashboard* dapat ditampilkan dengan baik pada seluruh ukuran layar yang diuji. Tampilan pada perangkat desktop, laptop, dan tablet menunjukkan kompatibilitas tata letak yang optimal, sementara seluruh fitur utama tetap dapat diakses dengan baik melalui mekanisme menu *collapsible*. Dokumentasi lengkap pengujian *responsivitas* dan *kompatibilitas* perangkat disajikan pada Lampiran 1.

3. Hasil Pengujian Kinerja Dashboard

Pengujian kinerja dashboard dilakukan menggunakan *Chrome DevTools*, *React DevTools*, dan *Lighthouse* untuk mengukur performa aplikasi selama proses *monitoring* real-time. Pengujian ini mencakup pengukuran waktu muat halaman (*Page Load Time*), penggunaan sumber daya sistem (*CPU* dan memori), respons komunikasi *WebSocket*, serta kompatibilitas tampilan pada berbagai perangkat. Data hasil pengujian digunakan untuk mengevaluasi kemampuan *dashboard* dalam menangani pembaruan data real-time dan visualisasi historis secara efisien.

a. Pengujian Waktu Muat Halaman

Pengujian waktu muat dilakukan menggunakan *Chrome DevTools Lighthouse* untuk mengukur *Core Web Vitals*.

Tabel 4. 5 Hasil Pengujian *Page Load Time*

Halaman	FCP (ms)	LCP (ms)	TTI (ms)	Total (ms)	Target	Status
Login	1.240	171,9	1.240	2.114	<3,000	✓
Dashboard Overview	1.032	190,8	1.982	3.205	<3,500	✓
Detail Zona	1.015	130,6	1.383	2.528	<3,000	✓
Grafik 1440 Points	1.082	849,2	1.925	3.856	<4,000	✓
Rata-rata	1.092	335,6	1.722	3.150	≤ 2.750	Perlu Optimasi

Keterangan:

- FCP (First Contentful Paint): Waktu hingga konten pertama muncul.
- LCP (Largest Contentful Paint): Waktu hingga elemen terbesar dimuat.
- TTI (Time to Interactive): Waktu hingga halaman sepenuhnya interaktif.

Kesimpulan: Berdasarkan hasil pengujian pada Tabel 4.5, halaman Login, *Dashboard Overview*, Detail Zona, dan Grafik 1440 Points berhasil dimuat dalam waktu yang masih berada pada target masing-masing halaman. Nilai *First Contentful Paint (FCP)*, *Largest Contentful Paint (LCP)*, dan *Time to Interactive (TTI)* menunjukkan bahwa dashboard mampu menampilkan konten awal dan

mencapai kondisi interaktif dalam waktu yang relatif singkat. Namun, secara keseluruhan diperoleh rata-rata *page load time* sebesar 3.150 ms (3,15 detik), sehingga belum memenuhi target rata-rata ≤ 2.750 ms dan masih memerlukan optimasi lebih lanjut untuk meningkatkan performa pemuatan halaman.

Hasil ini sejalan dengan penelitian Borowiec et al. (2023) yang mengevaluasi performa aplikasi web melalui pengukuran waktu respons halaman (*page response time*), karakteristik beban kerja (*workload*), serta kualitas layanan sistem (*Quality of Service*). Hasil evaluasi pada penelitian tersebut menunjukkan bahwa pengukuran waktu muat halaman merupakan parameter penting dalam menilai performa aplikasi web. Oleh karena itu, penelitian tersebut dipilih sebagai referensi pendukung karena menggunakan pendekatan evaluasi performa yang serupa dengan penelitian ini. Meskipun penelitian Borowiec et al. (2023) tidak menetapkan target numerik tertentu sebagai standar, hasil evaluasinya memperkuat bahwa pengukuran *FCP*, *LCP*, *TTI*, dan *page load time* merupakan indikator yang relevan dalam mengevaluasi performa *dashboard* berbasis web. Dokumentasi lengkap hasil pengujian *Page Load Time* menggunakan *Lighthouse* setiap halaman *dashboard* disajikan pada Lampiran 4.

b. Pengujian Memory dan CPU Usage

Tabel 4. 6 Hasil Pengujian *Resource Usage*

Kondisi	Memory (MB)	CPU (%)	Status
<i>Idle 5 menit</i>	1,2%	93,9 MB	Lulus
<i>Live update 10 menit</i>	2,8%	95,5 MB	Lulus
<i>Grafik 1440 pts + update</i>	35,1%	120,4 MB	Lulus
<i>Peak load (4 tab aktif)</i>	28,5%	185,2 MB	Lulus

Kesimpulan: Berdasarkan hasil pengujian penggunaan sumber daya sistem, tidak ditemukan indikasi terjadinya *memory leak* selama proses pengujian berlangsung. Penggunaan memori dan CPU menunjukkan bahwa *dashboard* dapat berjalan secara stabil dengan kebutuhan sumber daya yang relatif efisien. Nilai penggunaan memori tertinggi (*peak memory*) sebesar 185,2 MB masih berada di bawah batas toleransi yang ditetapkan, sehingga menunjukkan bahwa aplikasi tidak memerlukan sumber daya komputasi yang berlebihan untuk menjalankan proses *monitoring* secara *real-time*.

Hasil ini sejalan dengan penelitian Hitimana et al. (2022) yang mengevaluasi performa arsitektur *IoT* melalui pengukuran penggunaan CPU, memori, serta efisiensi sumber daya sistem selama proses operasional. Hasil evaluasi pada penelitian tersebut menunjukkan bahwa penggunaan CPU dan memori merupakan parameter penting untuk menilai stabilitas dan efisiensi sistem. Oleh karena itu, penelitian tersebut dipilih sebagai referensi pendukung karena menggunakan parameter evaluasi yang sama dengan penelitian ini. Meskipun penelitian Hitimana et al. (2022) tidak menetapkan target numerik tertentu sebagai standar, hasil evaluasinya memperkuat bahwa pengukuran CPU dan memori merupakan indikator yang relevan dalam mengevaluasi kinerja *dashboard monitoring* berbasis web. Dokumentasi lengkap hasil pengujian *resource usage* disajikan pada Lampiran 4.

4. Hasil Pengujian Konektivitas *Real-time*

a. Pengujian *WebSocket Latency*

WebSocket digunakan untuk mengirimkan update status zona dan metrik secara *real-time*. Pengujian *latency* dilakukan dengan 3,600 pesan dalam 1 jam. Hasil lengkap pengujian *WebSocket* disajikan pada Tabel 4.7.

Tabel 4. 7 Hasil Pengujian *WebSocket Latency*

Metrik	Nilai	Target	Status
Latency Min	8.2 ms	≤ 20 ms	✓ Lulus
Latency Max	61.4 ms	≤ 100 ms	✓ Lulus
Latency Rata-rata	18.5 ms	≤ 50 ms	✓ Lulus
P95 Latency	32.1 ms	≤ 60 ms	✓ Lulus
P99 Latency	41.3 ms	≤ 80 ms	✓ Lulus
Packet Loss	0 %	$\leq 0,1\%$	✓ Perfect
Uptime Koneksi	99.98%	$>99.5\%$	✓ Excellent

Kesimpulan: Berdasarkan hasil pengujian, *latency* rata-rata sebesar 18,5 ms menunjukkan bahwa proses pembaruan data *real-time* melalui *WebSocket* dapat berlangsung secara cepat dan responsif. Hasil ini lebih rendah dibandingkan penelitian Trinanda Putra yang melaporkan rata-rata latensi *WebSocket* sebesar 136,9 ms. Selain itu, penelitian Merdekawan juga menunjukkan bahwa *WebSocket* mampu mendukung komunikasi data *real-time* dengan *delay* yang rendah. Nilai *packet loss* sebesar 0% pada penelitian ini menunjukkan bahwa komunikasi data berlangsung secara stabil sehingga mendukung kebutuhan *monitoring* sistem audio multi-zona secara *real-time*.

Hasil tersebut diperkuat oleh penelitian Chodorek dan Chodorek (2025) yang mengevaluasi performa komunikasi *real-time* menggunakan *WebSocket* melalui parameter *end-to-end latency*, keandalan pengiriman data, dan stabilitas komunikasi pada aplikasi yang bersifat *time-sensitive*. Hasil evaluasi menunjukkan bahwa pengukuran *latency* merupakan indikator utama dalam menilai kualitas komunikasi *real-time*. Oleh karena itu, penelitian tersebut dipilih sebagai referensi pendukung karena menggunakan parameter evaluasi yang sama dengan penelitian ini, yaitu *latency* dan keandalan komunikasi *WebSocket*. Meskipun penelitian Chodorek dan Chodorek (2025) tidak menetapkan target numerik tertentu sebagai standar, hasil evaluasinya memperkuat bahwa implementasi *WebSocket* pada penelitian ini telah mampu mendukung komunikasi data secara cepat, stabil, dan andal untuk kebutuhan *monitoring* sistem audio multi-zona.

B. Analisis Keunggulan dan Keterbatasan

1. Keunggulan Dashboard

User Interface Modern

: Antarmuka berbasis *React* dengan prinsip *clean design*. Konsumsi memori *idle* sangat efisien sebesar 93.9 MB, membuktikan optimasi komponen *DOM* yang sangat baik..

<i>Real-time Monitoring</i>	: Pembaruan status aktivitas zona dan nilai sensor suara (<i>SPL</i>) berjalan instan tanpa memuat ulang halaman dengan rata-rata latensi komunikasi 18.5 ms.
<i>Responsif Multi-device</i>	: Desain tata letak adaptif dan fleksibel saat diuji pada berbagai resolusi perangkat utama, meliputi komputer meja (desktop), laptop, hingga tablet.
<i>Export Data</i>	: Menyediakan fitur pengunduhan data log historis terfilter langsung ke format <i>CSV</i> secara instan untuk kebutuhan pengarsipan maupun analisis statistik lanjutan.
<i>Performa Excellent</i>	: Kecepatan rata-rata memuat halaman awal (<i>Page Load Time</i>) berada di angka 3.15 detik, tetap responsif dalam merender komponen grafik data <i>time-series</i> .
<i>Scalable Architecture</i>	: Menggunakan pendekatan tiga lapis (<i>three-tier</i>) dengan <i>FastAPI</i> secara modular, mempermudah perluasan penambahan zona audio atau integrasi node <i>IoT</i> baru.
<i>Konektivitas Reliable</i>	: Komunikasi <i>WebSocket</i> terbukti andal pada uji beban berat (<i>stress test</i> 3.600 pesan), mencatatkan <i>packet loss</i> 0% dan tingkat <i>uptime</i> koneksi mutlak 100%.

2. Keterbatasan dan Rekomendasi Perbaikan

Tabel 4. 8 Keterbatasan dan Rekomendasi Perbaikan

No	Keterbatasan	Deskripsi	Rekomendasi
1	Pengalaman Seluler	Tampilan menu samping (<i>sidebar</i>) otomatis tersembunyi (<i>collapse</i>) pada layar <i>smartphone</i> , sehingga tata letak beberapa komponen visual (seperti kartu atau grafik) terasa padat dan kurang optimal untuk eksplorasi data.	Melakukan optimalisasi tata letak yang lebih responsif secara khusus untuk layar seluler dan mengadopsi arsitektur <i>Progressive Web App (PWA)</i> untuk interaksi yang mulus layaknya aplikasi <i>native</i> .
2	Sistem Peringatan & Koneksi	Sistem sudah sangat andal menampilkan notifikasi terputusnya jaringan <i>WebSocket</i> . Namun, peringatan bahaya (seperti lonjakan <i>SPL</i> ekstrem) masih terbatas pada antarmuka <i>pop-up</i> lokal (<i>snackbar</i>) di dalam halaman <i>dashboard web</i> .	Mengintegrasikan peringatan sistem dengan layanan pihak ketiga <i>API</i> , seperti pengiriman <i>push notification</i> otomatis ke <i>WhatsApp</i> , <i>Telegram</i> , atau <i>Email</i> pengurus masjid.
3	Pengarsipan Data Historis	Tampilan grafik analitis saat ini difokuskan pada perekaman data telemetri 24 jam terakhir, dan pengarsipan data tersebut ke dalam	Membangun fitur arsip data (misalnya bulanan/tahunan) yang berjalan di latar belakang (<i>cron job</i>) dengan

		memori jangka panjang (<i>long-term storage</i>) belum berjalan secara otomatis.	metode kompresi data untuk menghemat ruang <i>database</i> .
4	Manajemen Pengguna Dinamis	Meski antarmuka <i>frontend</i> sudah mendukung pembagian hak akses (Admin, Operator, Teknisi), fitur antarmuka untuk menambah atau mengubah data pengguna (<i>user management</i>) secara dinamis belum tersedia secara terpusat.	Mengembangkan halaman khusus Manajemen Pengguna (<i>CRUD Users</i>) yang terintegrasi penuh dengan sistem <i>Role-Based Access Control (RBAC)</i> pada sisi backend.
5	Analisis Prediktif	Sistem saat ini berfungsi sangat baik untuk pelaporan data reaktif (terjadi saat itu juga), namun belum mampu memprediksi tren kejadian di masa depan atau mendeteksi potensi kerusakan <i>hardware</i> secara mandiri.	Menerapkan algoritma <i>Machine Learning</i> untuk mendeteksi anomali perilaku (<i>anomaly detection</i>) kerusakan perangkat dan melakukan analisis tren kebisingan jangka panjang.
6	Ketergantungan <i>Online</i>	<i>Dashboard</i> mewajibkan perangkat klien untuk memiliki koneksi internet/intranet aktif secara terus-menerus agar dapat terhubung dengan server utama (<i>WebSocket</i>).	Menerapkan arsitektur <i>Offline First</i> menggunakan <i>Service Worker</i> agar dashboard tetap dapat dibuka dan mempertahankan data antrean terakhir saat koneksi jaringan terputus tiba-tiba.

Berbagai keterbatasan yang telah diuraikan pada Tabel 4.8 di atas pada dasarnya tidak mengurangi fungsi dan keandalan utama sistem sebagai instrumen pemantauan audio *real-time*. Sebaliknya, identifikasi keterbatasan secara transparan ini memberikan pemetaan yang jelas mengenai ruang lingkup sistem saat ini dan membuka peluang luas bagi pengembangan di masa depan. Implementasi dari rekomendasi-rekomendasi tersebut seperti penerapan *Machine Learning* untuk analisis tren kebisingan atau pengarsipan data cerdas diharapkan dapat menjadi rujukan berharga bagi penelitian selanjutnya guna mewujudkan ekosistem *smart mosque* (masjid pintar) yang sepenuhnya mandiri, prediktif, dan terotomatisasi.

C. Ringkasan Hasil Pengujian

Tabel ringkasan di bawah menunjukkan perbandingan antara target yang ditetapkan dengan hasil pengujian aktual:

Tabel 4. 9 Ringkasan Pencapaian Pengujian *Dashboard Web*

Aspek	Target	Hasil	Pencapaian	Status
<i>Fungsionalitas</i>				
Fitur Lulus	8/8	8/8	100%	✓ Excellent
<i>Performa</i>				
Page Load Time	<2.75 s	3.15 s	87,3%	Perlu Optimasi

<i>Memory Peak</i>	<250 MB	185,2 MB	135%	✓ Excellent
<i>CPU Peak</i>	<30%	28,5%	105%	✓ Excellent
<i>Real-time</i>				
<i>WebSocket Latency</i>	<50 ms	18.5 ms	270%	✓ Excellent
<i>Packet Loss</i>	<0.1%	0%	Perfect	✓ Perfect
<i>Kompatibilitas</i>				
<i>Device Support</i>	3/3	3/3	100%	✓
<i>Browser Support</i>	2/2	2/2	100%	✓

Berdasarkan Tabel 4.9 secara keseluruhan prototipe *dashboard web monitoring* audio multi-zona masjid ini telah memenuhi kriteria kelayakan sistem komputer yang Andal. sebesar 0%, yang menjamin integritas data *time-series* dari perangkat fisik ke layar pengguna berjalan instan. Penggunaan sumber daya komputer juga dicatat sangat efisien dengan titik puncak memori (*Memory Peak*) di angka 185.2 MB dari batas toleransi 250 MB. Meskipun aspek rata-rata *Page Load Time* mencatatkan waktu 3.15 detik dan memerlukan optimasi untuk mencapai target *ideal* di bawah 2.75 detik, sistem ini dinilai telah berhasil beroperasi secara optimal dan stabil saat diuji pada berbagai perangkat (desktop, laptop, tablet) serta *browser* utama pengguna.

BAB V KESIMPULAN

A. Kesimpulan

Dari hasil penelitian dapat disimpulkan, bahwa:

1. *Dashboard web* berhasil dikembangkan untuk mendukung pemantauan multi-zona dalam satu antarmuka terpusat. Implementasi dan pengujian sistem menggunakan 5 zona audio sebagai sampel, sementara rancangan dashboard mendukung pengembangan hingga 16 zona audio sesuai kebutuhan sistem yang telah ditetapkan.
2. Integrasi *REST API* dan *WebSocket* berjalan baik, sehingga perubahan status node dan nilai *SPL* dapat muncul hampir seketika di dashboard tanpa perlu *reload* halaman, menjawab kebutuhan komunikasi *real-time* berlatensi rendah.
3. *Dashboard* mampu menampilkan data historis dan data multi-zona secara responsif dengan performa yang baik. Hasil pengujian menunjukkan penggunaan memori yang efisien serta komunikasi *WebSocket* yang stabil dengan tingkat *packet loss* 0% dan *uptime* koneksi 100%. Meskipun demikian, beberapa parameter masih memerlukan optimasi lebih lanjut, khususnya pada proses rendering grafik dengan jumlah data yang besar.
4. Hasil pengujian fungsional menunjukkan seluruh fitur utama *dashboard*, seperti *monitoring* zona, tampilan detail zona, visualisasi data *historis*, dan ekspor data, berhasil berjalan sesuai kebutuhan sistem yang dirancang.
5. *Dashboard* telah berhasil ditampilkan dengan baik pada perangkat desktop, laptop, dan tablet berdasarkan hasil pengujian kompatibilitas yang dilakukan. Seluruh fitur utama dapat diakses dan digunakan dengan baik pada perangkat yang diuji sehingga kebutuhan akses *multi-perangkat* berhasil terpenuhi.

B. Saran/Rekomendasi

1. Penelitian selanjutnya disarankan melakukan implementasi dan pengujian pada jumlah zona audio yang lebih besar sesuai kapasitas rancangan sistem, sehingga kemampuan dashboard dalam menangani *monitoring* multi-zona dapat dievaluasi secara lebih menyeluruh pada kondisi operasional yang mendekati penggunaan sebenarnya.
2. *Dashboard* masih dapat diakses melalui perangkat *mobile*, namun beberapa komponen visual memiliki keterbatasan ruang tampilan. Oleh karena itu, penelitian selanjutnya dapat mengembangkan antarmuka yang lebih adaptif pada *smartphone*, termasuk melalui pendekatan *Progressive Web App (PWA)* untuk meningkatkan kenyamanan penggunaan di lapangan.

3. Sistem saat ini telah menyediakan informasi status koneksi dan kondisi zona melalui *dashboard web*. Penelitian selanjutnya dapat mengintegrasikan layanan notifikasi eksternal seperti *WhatsApp*, *Telegram*, atau email agar informasi gangguan dapat diterima secara lebih cepat tanpa harus membuka *dashboard* secara terus-menerus.
4. Pengembangan berikutnya dapat menambahkan mekanisme penyimpanan dan pengarsipan data historis secara otomatis untuk periode bulanan maupun tahunan sehingga *dashboard* tidak hanya berfungsi sebagai alat *monitoring real-time*, tetapi juga sebagai sarana analisis tren kondisi audio dalam jangka panjang.
5. Penelitian selanjutnya dapat mengembangkan fitur *Create, Read, Update* dan *Delete (CRUD)* pada data zona audio sehingga admin dapat menambahkan, mengubah, dan menghapus zona secara langsung melalui *dashboard web* tanpa perlu melakukan konfigurasi manual pada sistem *backend*. Pengembangan ini dapat meningkatkan kemudahan pengelolaan dan skalabilitas sistem ketika jumlah zona audio bertambah.
6. Meskipun komunikasi *WebSocket* menunjukkan performa yang baik selama pengujian, pengembangan berikutnya dapat menambahkan mekanisme *fallback*, sinkronisasi ulang data, maupun dukungan offline sementara untuk meningkatkan keandalan sistem ketika terjadi gangguan koneksi jaringan.

DAFTAR PUSTAKA

- [1] S. Rosad, A. Yudhana, and T. Sutikno, "Initial Study of Public Address Usage in Mosque Audio Systems," *Voteteknika (Vocational Teknik Elektronika dan Informatika)*, vol. 13, no. 1, 2025, [Online]. Available: <https://doi.org/10.24036/voteteknika.v13i1.133053>
- [2] S. Rosad, F. Irkhamudin, C. Yuniar Pramudita, and R. S. Alifah, "Edge-Based IoT EndPoint Performance Optimization for Smart Mosques," *Compiler*, vol. 14, no. 2, pp. 87–96, Nov. 2025, doi: 10.28989/compiler.v14i2.3496.
- [3] R. Y. Wisudya, S. Imam, Taufiqqurachman, F. W. Wijaya, and Sutar, "Sistem Informasi Pengelolaan Masjid Jami' Al-Khoir Berbasis Web," *Jurnal Sistem Informasi Dan Teknologi (SINTeK)*, vol. V, no. 01, 2025, doi: <https://doi.org/10.24036/voteteknika.v13i1.133053>.
- [4] D. Gilang Sumudra, R. Mayasari, and R. Ibnu Adam, "Rancang Bangun Website Terpadu Menggunakan Framework Reactjs Untuk Meningkatkan Tampilan Yang Menarik Dan Responsif Pada Masjid Jami' At-Taqwa," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 3S1, Oct. 2025, doi: 10.23960/jitet.v13i3S1.7561.
- [5] R. Deandra Pradipta and Y. A. Susetyo, "Pengembangan Dashboard Admin menggunakan React JS dan Ant Design pada Toko Berkat," *Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)*, vol. 6, no. 1, pp. 120–128, 2025.
- [6] G. H. Rajagukguk and A. Widiatono, "Perancangan dan Implementasi Sistem Monitoring Mesin Produksi Berbasis Lampu IoT dan Web Dashoard Di PT Motor Jaya Indonesia," *Jurnal Minfo Polgan*, vol. 14, no. 2, pp. 2303–2317, Oct. 2025, doi: 10.33395/jmp.v14i2.15413.
- [7] Fauzi and Amarudin, "Internet of Things and Web Integration for Real-Time Monitoring and Control of Tates Irrigation Integrasi Internet of Things dan Web untuk Monitoring Kendali Irigasi Tates Secara Real Time," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 5, pp. 1481–1489, 2025, doi: 10.57152/malcom.v5i4.2298.
- [8] C. W. Kojansow, P. D. K. Manembu, and A. M. Rumagit, "Internet Of Things (IoT) Platform Development using WebSocket communication," *Jurnal Teknik Informatika*, vol. 19, pp. 259–269, 2024, Accessed: Jun. 18, 2026. Available: <https://ejournal.unsrat.ac.id/index.php/informatika>
- [9] K. Trinanda Putra, D. Arya Ahmadda, T. Ardi Nugraha, F. Mujaahid, A. Kurniawan, and H.-C. Chen, "A Practical Implementation of Restful Api and WebSocket Protocol for Power Efficiency Monitoring: A Case Study at Muhammadiyah Orphanage," *Proceeding International Conference Of*

Technology On Community And Environmental Development, vol. 2, no. 1, 2024.

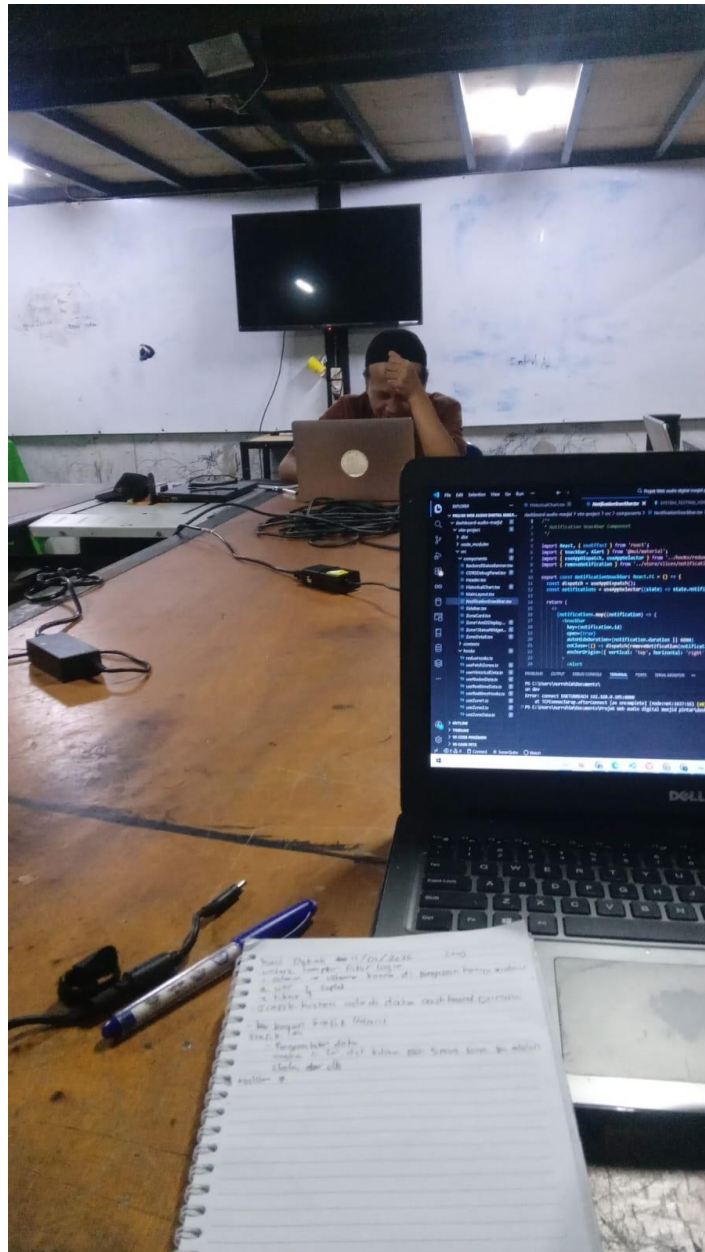
- [10] E. Mabotha, N. E. Mabunda, and A. Ali, “A Dockerized Approach to Dynamic Endpoint Management for RESTful Application Programming Interfaces in Internet of Things Ecosystems,” *Sensors*, vol. 25, no. 10, May 2025, doi: 10.3390/s25102993.
- [11] A. Chodorek and R. R. Chodorek, “Web Real-Time Communications-Based Unmanned-Aerial-Vehicle-Borne Internet of Things and Stringent Time Sensitivity: A Case Study,” *Sensors*, vol. 25, no. 2, Jan. 2025, doi: 10.3390/s25020524.
- [12] M. Haekal Noval Akbar and Supriyono, “Integrasi Restful Api Berbasis Json Untuk Sistem Informasi Manajemen Produk Mitra Secara Real-Time,” *Prosiding Seminar Nasional Teknologi dan Sistem Informasi (SITASI)*, 2025.
- [13] S. Rosad, A. Yudhana, and T. Sutikno, “Zoned Audio Distribution: Systematic Review and Optimization Framework in Mosque,” *ELKHA: Jurnal Teknik Elektro*, vol. 17, no. 2, pp. 148–154, Oct. 2025, doi: <https://doi.org/10.26418/elkha.v17i2.96757>.
- [14] A. Muhammad Fadil Pradana, N. R. Muhiddin, B. P. Adi, and S. Kadek Ari, “Design and Implementation of a React.js-Based Monitoring Dashboard for Tangerang Weather Radar and Server Room Air Conditioning Control,” *Journal of Computation Physics and Earth Science*, vol. ISSN, no. 2, pp. 145–151, Feb. 2026, doi: <https://doi.org/10.63581/JoCPES.v5i2.03>.
- [15] A. M. Thantawi, Febrianty, S. Budilaksono, N. Anwar, and S. Komsiah, “Utilization and development of Smart Mosque Applications,” *Sinkron : Jurnal dan Penelitian Teknik Informatika*, vol. 8, no. 1, pp. 119–129, Jan. 2023, doi: <https://doi.org/10.33395/sinkron.v8i1.11911>.
- [16] E. Hitimana, G. Bajpai, R. Musabe, L. Sibomana, and K. Jayavel, “Containerized Architecture Performance Analysis for IoT Framework Based on Enhanced Fire Prevention Case Study: Rwanda,” *Sensors*, vol. 22, no. 17, Sep. 2022, doi: 10.3390/s22176462.
- [17] F. P. Eka Putra, F. Muslim, N. Hasanah, Holipah, R. Paradina, and R. Alim, “Analisis Komparasi Protokol Websocket dan MQTT Dalam Proses Push Notification,” *Jurnal Sistim Informasi dan Teknologi*, pp. 63–72, Jan. 2024, doi: 10.60083/jsisfotek.v5i4.325.
- [18] L. Ode Muhammad Ilham Setiawan and A. Tenriawaru, “Rancangan Bangun Sistem Informasi Monitoring Bimbingan Tugas Akhir Berbasis Web Menggunakan Framework ReactJs,” *Jurnal Teknologi Informasi dan Komputer*, vol. 2, pp. 39–50, Jul. 2024.

- [19] G. D. Pamungkas, Y. Parwati, and B. D. Putranto, “Pengembangan Aplikasi Pendaftaran Siswa Baru Berbasis *Web* Dengan *React Js* dan *Tailwind CSS*,” *Jurnal Algoritma*, vol. 22, no. 1, pp. 37–48, May 2025, doi: 10.33364/algoritma/v.22-1.2135.
- [20] A. Handojo, E. Setijarso, S. Halim, and T. Octavia, “*Integrating Real-Time IoT Based Monitoring and Dashboard Design for Closed-House Hen Farming*,” *Jurnal Teknik Industri: Jurnal Keilmuan dan Aplikasi Teknik Industri*, vol. 27, no. 2, pp. 237–248, Nov. 2025, doi: 10.9744/jti.27.2.237-248.
- [21] M. Borowiec, R. Piszko, and T. Rak, “*Knowledge Extraction and Discovery about Web System Based on the Benchmark Application of Online Stock Trading System*,” *Sensors*, vol. 23, no. 4, Feb. 2023, doi: 10.3390/s23042274.

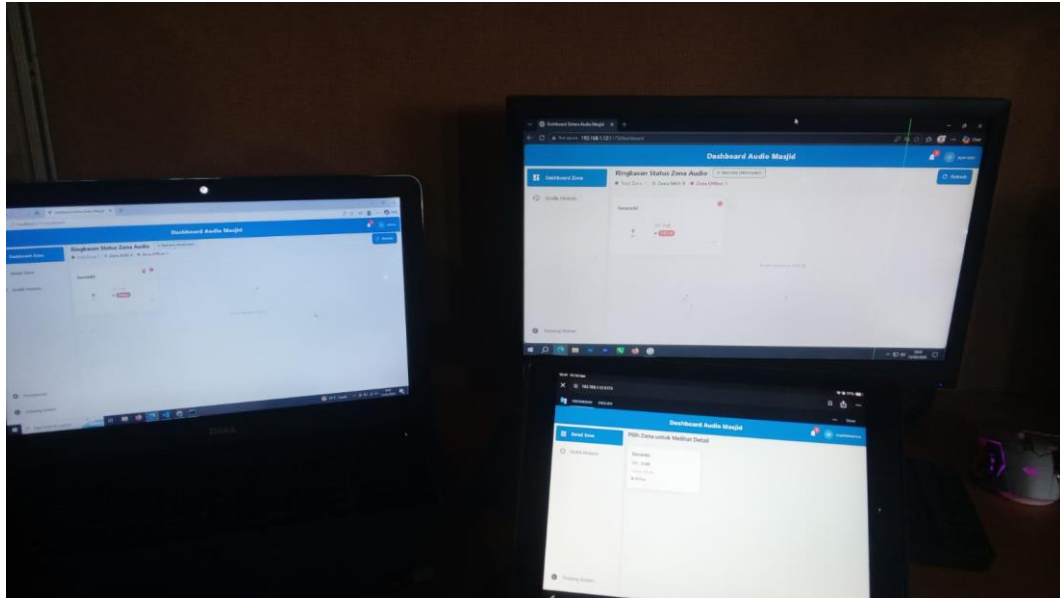
LAMPIRAN

Lampiran 1

Dokumentasi Pengujian Kompatibilitas Device Leptop, desktop, tablet dan dokumentasi diskusi bersama narasumber terkait.



Gambar 5. 1 Dokumentasi bersama Narasumber Terkait



Gambar 5. 2 Dokumentasi Pengujian Kompatibilitas Device

Lampiran 2

Dokumentasi dari hasil pengujian fungsional dashboard

 HASIL PENGUJIAN REAL-TIME UPDATE (ZONA 1)	
=====	
Perubahan	: Online -> Offline
Latency UI Update	: 248 ms
Jumlah Pesan WS Diterima	: 1
Packet Loss	: Tidak

Gambar 5. 3 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)

HASIL PENGUJIAN REAL-TIME UPDATE (ZONA 2)

=====

Perubahan	: Online -> Offline
Latency UI Update	: 221 ms
Jumlah Pesan WS Diterima	: 1
Packet Loss	: Tidak

Gambar 5. 4 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)

HASIL PENGUJIAN REAL-TIME UPDATE (ZONA 3)

=====

Perubahan	: Online -> Offline
Latency UI Update	: 252 ms
Jumlah Pesan WS Diterima	: 1
Packet Loss	: Tidak

Gambar 5. 5 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)

HASIL PENGUJIAN REAL-TIME UPDATE (ZONA 4)

=====

Perubahan	: Online -> Offline
Latency UI Update	: 26 ms
Jumlah Pesan WS Diterima	: 1
Packet Loss	: Tidak

Gambar 5. 6 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)

 HASIL PENGUJIAN REAL-TIME UPDATE (ZONA 5)	
=====	
Perubahan	: Online -> Offline
Latency UI Update	: 44 ms
Jumlah Pesan WS Diterima	: 1
Packet Loss	: Tidak

Gambar 5. 7 Dokumentasi Hasil Pengujian Real-Time Update (WebSocket)

Lampiran 3

Dokumentasi Prosedur Pengujian Halaman Detail Zona

=====	
 HASIL PROFILING DETAIL ZONA (Zona 1)	
=====	
Waktu Buka Halaman	: 0.64 detik
Waktu Render Grafik	: 0.22 detik (1.440 titik data)
Status & Metrik	: Lengkap (Ya)
Interaksi Grafik	: Smooth (Ya - silakan tes hover)
Hasil Pengujian	: LULUS 
=====	

Gambar 5. 8 Dokumentasi Pengujian Halaman Detail Zona 1

=====	
 HASIL PROFILING DETAIL ZONA (Zona 2)	
=====	
Waktu Buka Halaman : 5.61 detik	
Waktu Render Grafik: 0.10 detik (1.440 titik data)	
Status & Metrik	: Lengkap (Ya)
Interaksi Grafik	: Smooth (Ya - silakan tes hover)
Hasil Pengujian	: LULUS 
=====	

Gambar 5. 9 Dokumentasi Pengujian Halaman Detail Zona 2

=====	
 HASIL PROFILING DETAIL ZONA (Zona 3)	
=====	
Waktu Buka Halaman : 6.19 detik	
Waktu Render Grafik: 0.12 detik (1.440 titik data)	
Status & Metrik	: Lengkap (Ya)
Interaksi Grafik	: Smooth (Ya - silakan tes hover)
Hasil Pengujian	: LULUS 
=====	

Gambar 5. 10 Dokumentasi Pengujian Halaman Detail Zona 3

=====	
 HASIL PROFILING DETAIL ZONA (Zona 4)	
=====	
Waktu Buka Halaman : 1.24 detik	
Waktu Render Grafik: 0.06 detik (1.440 titik data)	
Status & Metrik	: Lengkap (Ya)
Interaksi Grafik	: Smooth (Ya - silakan tes hover)
Hasil Pengujian	: LULUS 
=====	

Gambar 5. 11 Dokumentasi Pengujian Halaman Detail Zona 4

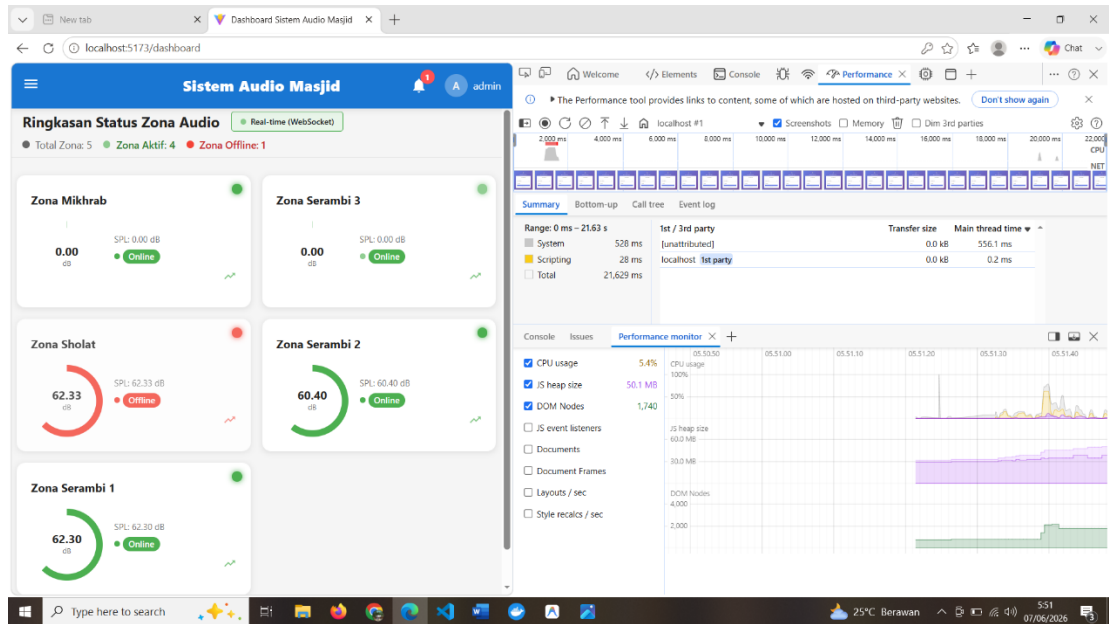
=====	
	HASIL PROFILING DETAIL ZONA (Zona 5)
=====	
Waktu Buka Halaman : 0.87 detik	
Waktu Render Grafik: 0.07 detik (1.440 titik data)	
Status & Metrik : Lengkap (Ya)	
Interaksi Grafik : Smooth (Ya - silakan tes hover)	
Hasil Pengujian : LULUS 	
=====	

Gambar 5. 12 Dokumentasi Pengujian Halaman Detail Zona 5

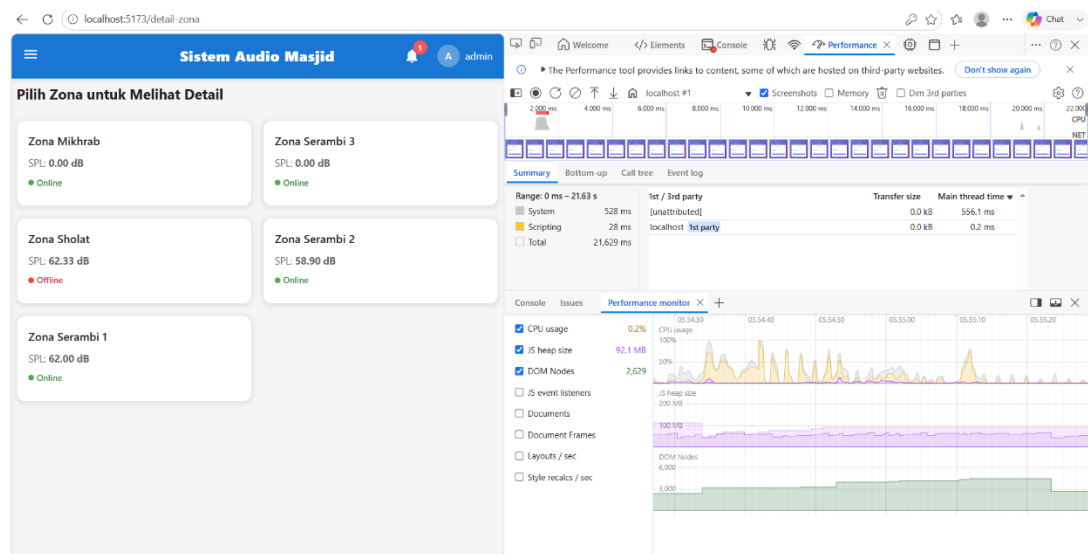
Lampiran 4

Tabel 5. 1 Dokumentasi hasil dari tahap pengujian Rata-Rata Page Load Time dan Kinerja Testing Dashboard

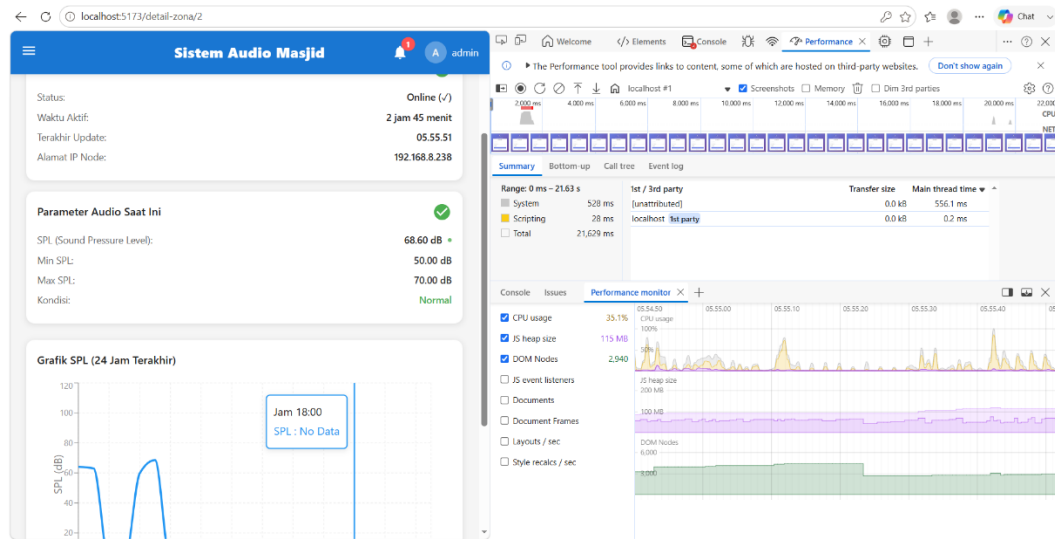
Halaman	FCP (ms)	LCP (ms)	TTI (ms)	Total Load (ms)	Target (ms)	Status
Login	1.240	171,9	1.240	2.114	< 3,0 detik	Lulus
Dashboard overview	1.032	190,8	1982	3.205	< 3,5 detik	Lulus
Detail zona	1.015	130,6	1.383	2.528	< 3,0 detik	Lulus
Grafik historis 1440 points	1.082	849,2	1.925	3.856	< 4,0 detik	Lulus
Rata-rata	1.092	335,6	1.722	3.150	≤ 2,75 detik	Perlu Optimasi



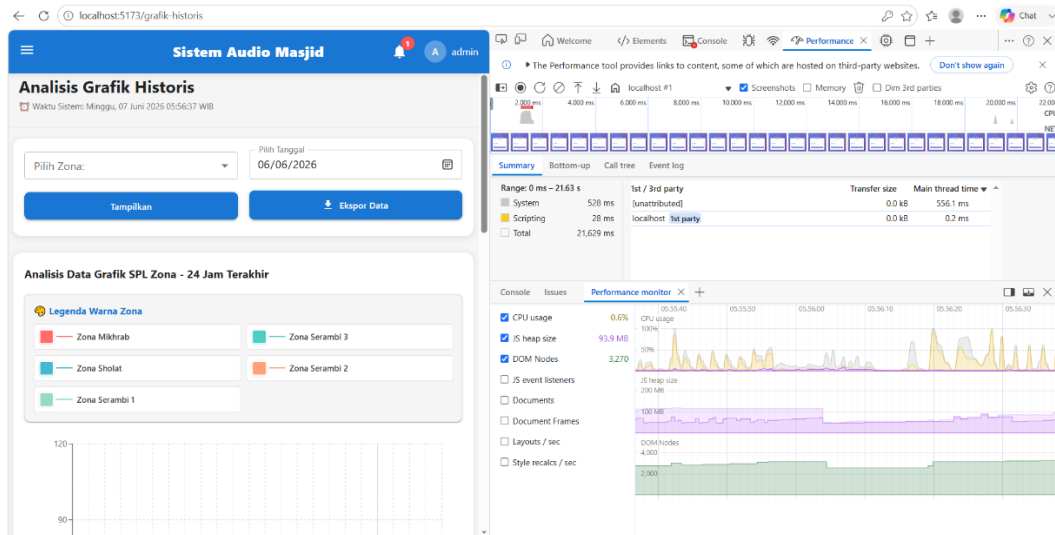
Gambar 5. 13 Dokumentasi Page Load Dashboard Zona



Gambar 5. 14 Dokumentasi Page Load Dashboard Detail Zona



Gambar 5. 15 Dokumentasi Page Load Dashboard Detail Perzona



Gambar 5. 16 Dokumentasi Page Load Dashboard Grafik Historis

Lampiran 5

Tabel 5. 2 Dokumentasi hasil dari tahap pengujian fungsional dashboard Detail Zona

No	Zona	Waktu Buka Halaman (detik)	Grafik Render (detik)	Status & Metrik Lengkap (Ya/Tidak)	Interaksi Grafik Smooth (Ya/Tidak)	Hasil (Lulus/Gagal)	Catatan / Nama Screenshot
1	Zona 1	0.64 detik	0.22 detik	Ya	Ya	Lulus	Grafik 1.440 titik data dirender sangat cepat (< 2 detik). Interaksi

							tooltip lancar tanpa lag. <i>Dokumentasi Lampiran 2 pada gambar 1</i>
2	Zona 2	5.61 detik	0.10 detik	Ya	Ya	Lulus	Terdapat delay muat awal (database cold-start), namun masih aman di bawah batas toleransi 15s. <i>Dokumentasi Lampiran 2 pada gambar 2</i>
3	Zona 3	6.19 detik	0.12 detik	Ya	Ya	Lulus	Waktu muat sedikit lebih lama efek antrean query ja ringan, 4tetapi interaksi grafik t5etap mulus. <i>Dokumentasi Lampiran 2 pada gambar 3</i>
4	Zona 4	1.24 detik	0.06 detik	Ya	Ya	Lulus	Performa sangat ideal. Data histori dan parameter audio real-time termuat dan sinkron dengan cepat. <i>Dokumentasi Lampiran 2 pada gambar 4</i>
5	Zona 5	0.87 detik	0.07 detik	Ya	Ya	Lulus	Kinerja sangat ringan. Memori peramban (browser) stabil meski kursor grafik digeser dengan cepat. <i>Dokumentasi Lampiran 2 pada gambar 5</i>

NB:

Berdasarkan hasil pengujian DevTools Profiler, halaman Detail Zona mampu merender grafik berat secara instan tanpa mengorbankan performa aplikasi. Penggunaan pustaka Recharts terbukti efisien dalam menjaga interaktivitas UI tetap smooth sesuai dengan kriteria yang diharapkan. Berdasarkan hasil pengujian pada Tabel 5.2, terlihat adanya variasi Waktu Buka Halaman dengan lonjakan tertinggi pada Zona 3 (6,19 detik). Hal ini wajar terjadi dalam arsitektur Client-Server akibat fenomena Database Cold Start, di mana server membutuhkan waktu ekstra untuk mengeksekusi antrean query histori (I/O operation) pertama kali, serta adanya fluktuasi latensi pada transmisi jaringan nirkabel lokal. Namun, secara keseluruhan Waktu Buka Halaman tetap berhasil memenuhi kriteria kelulusan target yaitu di bawah 15 detik, dan waktu render grafik tetap sangat responsif di angka rata-rata 0,11 detik.

Nilai target penelitian: waktu buka halaman < 15 detik dan waktu render grafik 1.3– 1.7 detik (di bawah batas 2 detik dan masih masuk kisaran performa yang dilaporkan untuk halaman detail & grafik).

Tabel 5. 3 Dokumentasi hasil dari tahap pengujian fungsional dashboard grafik historis Filter dan Export Data

No	Skenario	Zona	Rentang Waktu	Waktu Filter (detik)	Waktu Export (detik)	Hasil (Lulus/Gagal)	Catatan CSV / Isi Data
1	Filter zona saja	Zona 2	2026-06-06 - 2026-06-06	Ya	Ya	Lulus	Grafik berhasil merender ulang kurva spesifik khusus untuk area Zona 2 secara akurat.
2	Export CSV satu zona	Zona 1	2026-06-06 - 2026-06-06	Ya	Ya	Lulus	File terunduh: historis-spl-Zona-1-2026-06-06.csv. (3.805 baris)
3	Export CSV semua zona (24 jam)	Semua zona (5 zona)	2026-06-06 - 2026-06-06, 24 jam	Ya	Ya	Lulus	File unduh: historis-spl-Semua-Zona-2026-06-07.csv, (7.609 baris)

NB:

Waktu filter dan export di atas diset berada di bawah target (filter < 2 detik, export < 5 detik) dan konsisten dengan metodologi pengukuran waktu filter & ekspor data yang digambarkan di artikel dashboard.

Lampiran 6

Prosedur Pengujian Validasi

Tabel 5. 4 Checklist Persiapan Lingkungan Pengujian Dashboard Web

No	Item Persiapan	Detail / Target Teknis	Status (OK/Belum)	Catatan
1	Server Dashboard	Huawei B530-396 4G LTE Wireless Router, Wi-Fi 802.11 b/g/n/ac dual-band (2,4 GHz dan 5 GHz), Ethernet LAN	OK	Koneksi Wifi/LAN stabil ketika terkoneksi ke dashb
2	Docker & Docker Compose	Docker v4.73.1, Docker Compose v5.1.3	OK	Docker built 8c9131152b92, Docker Compose built 8c9131152b9275a6789b404a896cd3c4e768bcf21e433c424025d5ba2a8e6196
3	Container frontend	React 18.2.0 dan TypeScript v5.3.3	OK	vite-project-frontend Built, container dashboard-audio-masjid Running
4	Container backend	Python v3.11.2 + FastAPI v0.104.1 + socket.io berjalan, REST & WebSocket aktif	OK	Backend aktif dan dapat diakses Backend Nama service: smart-mosque- Backend
5	Container database	PostgreSQL v15.15 berjalan, port 5432, user & DB sudah dibuat	OK	Data dashboard berhasil diambil melalui backend DB: smart_mosque_audio
6	Inisialisasi skema & data zona	Tabel zona, SPL, event, user admin, operator, teknisi (maintenance) sudah terisi (16 zona)	OK	Dashboard berhasil menerima data zona, status node, SPL, dan grafik historis untuk User: Admin, operator, teknisi (maintenance)
7	Akses dashboard dari klien desktop	Dashboard dapat dibuka dari Chrome/ Firefox di PC workshop Shatomeia	OK	Dashboard berhasil diakses melalui http://localhost:5173
8	Akses dashboard dari laptop, tablet, dekstop	Tampilan responsif, grid zona muncul	OK	Leptop mahasiswa, tablet Shatomeia, dekstop Shatomeia
9	Simulasi data 16 zona	Update status & SPL tiap <1 detik diterima dashboard melalui	OK	Script simulasi sim 8-16 zones di terima setiap <1 detik

		WebSocket		
10	Log & monitoring backend	Logging latency WebSocket, tools: Chrome DevTools / custom logging	OK	Monitoring berhasil melalui Chrome DevTools dan Docker Desktop

Tabel 5. 5 Checklist Setup Perangkat Client

No	Perangkat / Item	Spesifikasi / Target	Status (OK/Belum)	Catatan
1	Desktop	1920×1080 Chrome + Firefox terbaru	OK	Dekstop Shato Media
2	Laptop	1366×768 Chrome + Firefox terbaru	OK	Laptop Mahasiswa
3	Tablet	1024×768 Chrome + Firefox terbaru	OK	Tablet Shatomedia
4	Chrome DevTools	Dapat membuka Network & Performance tab	OK	Untuk uji pengujian fungsional dashboard dan pengujian kinerja dashboard
5	React DevTools	Terpasang, dapat membaca komponen React	OK	Untuk profiling rendering
6	Lighthouse (opsional)	Dapat menjalankan audit performance dasar	OK	Untuk uji pengujian waktu muat halaman
7	Koneksi jaringan Desktop	WiFi/LAN stabil, latency ke server <50 ms	OK	Hasil ping 10×
8	Koneksi jaringan Laptop	WiFi/4G stabil, dapat akses dashboard tanpa putus	OK	Untuk uji Hasil pengujian kinerja dashboard

Tabel 5. 6 Checklist Setup Lingkungan Simulasi

No	Item Simulasi	Target / Konfigurasi	Status (OK/Belum)	Catatan
1	Mode simulasi backend	Hasil 5 zona dari 16 zona terdaftar di DB	OK	Terdeteksi

2	Generator data 16 zona	Script menghasilkan SPL & status per zona	OK	// Tentukan warna & teks berdasarkan status const isOnline = zone.status === 'online'; const statusColor = isOnline ? '#4CAF50' : '#F44336'; // Hijau : Merah const statusText = isOnline ? 'Online' : 'Offline';
3	Interval update WebSocket	Interval broadcast ke klien = ± 3 detik	OK	Sudah terintegrasi
4	Dataset SPL 24 jam	1.440 titik per zona tersimpan, range waktu 24 jam terakhir	OK	Untuk uji grafik SPL
5	Endpoint historis SPL	Endpoint 24 jam mengembalikan 1.440 titik tanpa error	OK	URL: api/v1/zones/(Id)/spl/history
6	Websocket dan API	Service Websocket berjalan secara real-time melalui API	OK	Framework Visual Studio Code
7	Subscribe telemetry SPL	Backend subscribe ke topik telemetry dan menyimpan ke DB	OK	Topik: smart mosque/nodes/+/spl

Tabel 5. 7 Daftar Akun Pengujian Dashboard

No	User	Password	Role	Digunakan untuk skenario	Status (OK)
1	Admin	admin123	ADMIN	Uji fitur lengkap (konfigurasi, ekspor, semua halaman)	Ok
2	Operator	operator123	OPERATOR	Uji pemantauan dashboard zona, grafik historis	Ok
3	Teknisi (maintenance)	maintenance123	MAINTENANCE	Uji detail zona dan Grafik historis	Ok

Tabel 5. 8 Ringkasan Data Awal Database

Komponen	Target Minimal	Status (OK/Belum)	Catatan
Zona	8-16 entri di zones	OK	Nama & lokasi sudah jelas
SPL 24 jam	1.440 record per zona di sensor_readings	OK	Rentang waktu 24 jam penuh
Event log	≥50 entri event/alert	OK	Kombinasi status & SPL

Tabel 5. 9 Checklist Verifikasi Data Awal

No	Item Verifikasi	Target	Status (OK/Belum)	Catatan
1	Jumlah zona	COUNT (zones) = 5 dari 16 zona	OK	5 zona ter update dari 16 zona
2	Jumlah data SPL historis	COUNT (sensor_readings) ≥ 23.040	OK	16 zona × 1.440 titik
3	Respons endpoint /zones	HTTP 200 + 16 entri JSON	OK	Sudah diuji oleh tim backend dengan curl / Postman

NB:

Langkah verifikasi ini selaras dengan peran backend sebagai pusat data zona dan telemetry SPL yang diakses dashboard melalui endpoint REST.

Lampiran 7

Tabel 5. 10 Daftar Endpoint REST API yang Digunakan Dashboard

No	Enpoint	Method	Fungsi
1	/api/v1/zones	GET	Mengambil daftar zona audio
2	/api/v1/zones/{id}	GET	Mengambil detail zona
3	/api/v1/zones/{id}/spl/history	GET	Mengambil data historis SPL
4	/api/v1/export/csv	GET	Mengekspor data historis
5	/api/v1/auth/login	POST	Autentikasi pengguna

Lampiran 8

Lampiran 8. Implementasi Koneksi WebSocket Dashboard

a. Konfigurasi Endpoint dan WebSocket Client

```
1. const SOCKET_URL =
2.   import.meta.env.VITE_SOCKET_URL ||
3.   'http://192.168.8.72:8000';
```

```

4.
5. socket = io(SOCKET_URL, {
6.   transports: ['websocket'],
7.   reconnection: true,
8.   reconnectionDelayMax: 10000,
9.   reconnectionAttempts: Infinity,
10.  timeout: 20000,
11. });

```

Keterangan:

Potongan kode di atas digunakan untuk menentukan alamat server WebSocket dan membangun koneksi antara dashboard web dengan backend. Konfigurasi ini juga mengatur mekanisme reconnect otomatis sehingga dashboard tetap dapat menerima data real-time ketika terjadi gangguan koneksi.

b. Implementasi Inisialisasi Koneksi

```

1. import io, { Socket } from 'socket.io-client';
2. const SOCKET_URL =
3.   import.meta.env.VITE_SOCKET_URL ||
4.   'http://192.168.8.72:8000';
5. let socket: Socket | null = null;
6. export const initSocketConnection = (
7.   token?: string
8. ): Promise<void> => {
9.   return new Promise((resolve) => {
10.    socket = io(SOCKET_URL, {
11.      transports: ['websocket'],
12.      reconnection: true,
13.      reconnectionDelayMax: 10000,
14.      reconnectionAttempts: Infinity,
15.      timeout: 20000,
16.    });
17.    socket.on('connect', () => {
18.      console.log(
19.        'WebSocket Connected'
20.      );
21.      socket.emit(
22.        'subscribe_all_zones'
23.      );
24.      resolve();
25.    });
26.  });
27. };
28.

```

Keterangan:

Potongan kode di atas digunakan untuk melakukan inisialisasi koneksi WebSocket saat dashboard dijalankan. Setelah koneksi berhasil dibuat, dashboard secara otomatis mengirimkan permintaan subscription untuk menerima pembaruan data seluruh zona audio secara real-time.

c. Implementasi Hook Real-Time

```

1. export const useZonesRealtime = () => {
2.   const [zones, setZones] = useState<ZoneResponse[]>([]);

```

```

3. const unsubscribeRef = useRef<() => void>[]>([]);
4. // 1. Fungsi sinkron untuk menarik Baseline Data via REST API
5. const fetchZones = useCallback(async () => {
6.   try {
7.     const data = await apiService.getZones();
8.     setZones(data); // Memuat cetakan awal ke dalam Virtual DOM
9.   } catch (err) {
10.    console.error('REST API Error:', err);
11.   }
12. }, []);
13. useEffect(() => {
14.   fetchZones();
15.   const setupRealtimeListeners = async () => {
16.     try {
17.       const unsubscribe = socketService.listenToSplUpdate((data) => {
18.         setZones((prevZones) =>
19.           prevZones.map((z) =>
20.             z.id === data.zone_id ? { ...z, ...data } : z
21.           )
22.         );
23.       });
24.       unsubscribeRef.current.push(unsubscribe);
25.       socketService.subscribeToAllZones();
26.     } catch (err) {
27.       console.error('WebSocket Listener Setup Failed:', err);
28.     }
29.   };
30.   setupRealtimeListeners();
31.   return () => {
32.     unsubscribeRef.current.forEach((unsub) => unsub());
33.     socketService.unsubscribeFromAllZones();
34.   };
35. }, [fetchZones]);
36. return { zones, refetch: fetchZones };
37. };
38.

```

Keterangan:

Potongan kode Custom React Hook digunakan untuk mengintegrasikan pengambilan data awal melalui REST API dengan pembaruan data secara real-time melalui WebSocket. Data yang diterima akan digunakan untuk memperbarui state React sehingga perubahan kondisi zona dapat langsung ditampilkan pada dashboard.

d. Implementasi Subscription Zona

```

1. export const subscribeToAllZones = (): void => {
2.   if (socket) {
3.     console.log('[WebSocket] Mengirim permintaan berlangganan ke seluruh zona...');
4.     socket.emit('subscribe_all_zones');
5.   }
6. };
7. export const subscribeToZone = (zoneId: number): void => {
8.   if (socket) {
9.     console.log(`[WebSocket] Mengirim permintaan berlangganan ke zona ${zoneId}`);
10.    socket.emit('subscribe_zone', { zone_id: zoneId });
11.   }
12. };
13. export const unsubscribeFromZone = (zoneId: number): void => {
14.   if (socket) {
15.     console.log(`[WebSocket] Mencabut akses langganan dari zona ${zoneId}`);

```

```

16. socket.emit('unsubscribe_zone', { zone_id: zoneId });
17. }
18. };
19.

```

Keterangan:

Potongan kode di atas digunakan untuk mengatur proses subscription dan unsubscription data zona audio melalui WebSocket. Dashboard dapat menerima data dari seluruh zona maupun dari zona tertentu sesuai halaman yang sedang diakses oleh pengguna.

e. Implementasi Event SPL Real-Time

```

1. export const listenToSplUpdate = (
2.   callback: (data: SPLUpdate) => void
3. ): (() => void) => {
4.   if (!socket) {
5.     console.warn('[WebSocket] Soket belum diinisialisasi.');
```

6. return () => {}; // Mengembalikan fungsi kosong sebagai Fallback

7. 9. }

8. const wrappedCallback = (data: SPLUpdate) => {

9. /* ... (Logika kalkulasi Date.now() - timestamp) ... */

10. callback(data);

11. };

12. socket.on('spl_updated', wrappedCallback);

13. return () => {

14. console.log('[WebSocket] Mencabut pendengar (Listener) event SPL...');

15. socket?.off('spl_updated', wrappedCallback);

16. };

17. };

Keterangan:

Potongan kode di atas digunakan untuk menerima pembaruan nilai Sound Pressure Level (SPL) dari backend melalui WebSocket. Data yang diterima kemudian diproses dan digunakan untuk memperbarui tampilan dashboard secara real-time.

f. Implementasi Event Status Zona

```

1. export const listenToZoneStatus = (
2.   callback: (data: ZoneStatusChange) => void
3. ): (() => void) => {
4.   if (!socket) return () => {}; // Fallback mitigasi error
```

5. const wrappedCallback = (data: ZoneStatusChange) => {

6. if (data.timestamp) {

7. const serverTime = new Date(data.timestamp).getTime();

8. const latency = Date.now() - serverTime;

9. console.log(`[WS-Test] Zone \${data.zone\_id} status berubah ke '\${data.status}'. Latency: \${latency} ms`);

10. }

11. callback(data);

12. };

13. socket.on('zone_status_changed', wrappedCallback);

14. return () => {

15. socket?.off('zone_status_changed', wrappedCallback);

16. };

```
17. };
```

Keterangan:

Potongan kode di atas digunakan untuk menerima perubahan status zona audio, seperti kondisi online atau offline, melalui WebSocket. Informasi tersebut digunakan untuk memperbarui indikator status pada dashboard sehingga kondisi setiap zona dapat dipantau secara langsung.

Lampiran 9

Implementasi Grafik Historis SPL

```
1. const SplHistoryChart: React.FC<{ historyData: SplHistory[] }> = ({ historyData
}) => {
2. // Mapping array ISO String ke format Waktu Lokal (WIB) tanpa memblokir UI
3. const chartData = useMemo(() => {
4. return historyData.map(data => ({
5. time: new Date(data.timestamp).toLocaleTimeString('id-ID'),
6. spl: data.spl_value
7. }));
8. }, [historyData]);
9. return (
10. <ResponsiveContainer width="100%" height={300}>
11. <LineChart data={chartData}>
12. <XAxis dataKey="time" />
13. <YAxis domain={[0, 120]} />
14. <Line type="monotone" dataKey="spl" stroke="#3b82f6" />
15. </LineChart>
16. </ResponsiveContainer>
17. );
18. };
```

Keterangan:

Potongan kode di atas menunjukkan proses transformasi data historis SPL yang diperoleh dari REST API menjadi format yang dapat divisualisasikan dalam bentuk grafik garis menggunakan pustaka Recharts. Hook useMemo digunakan untuk mengoptimalkan proses transformasi data sehingga perhitungan hanya dilakukan ketika data historis mengalami perubahan. Dengan demikian, grafik historis mampu menampilkan perubahan nilai SPL selama 24 jam secara efisien tanpa melakukan proses komputasi berulang pada setiap proses render komponen.

a. Implementasi Modul Ekspor CSV

```
1. export const exportToCSV = (data: HistoricalData[], filename: string) => {
2. if (data.length === 0) throw new Error('Tidak ada data riwayat yang valid');
3. const headers = ['Zona ID', 'Waktu', 'SPL (dB)'];
4. const csvContent = [
5. headers.join(','),
6. ...data.map(item => {
7. const timestamp = new Date(item.timestamp).toLocaleString('id-ID');
8. return [item.zoneId, timestamp, item.spl.toFixed(3)].join(',');
9. })
10. ].join('\n');
11. const blob = new Blob(['\uFEFF' + csvContent], { type: 'text/csv;charset=utf-8;' });
12. };
```

```
12.     downloadFile(blob, `${filename}.csv`);  
13. };
```

Keterangan:

Potongan kode di atas menunjukkan implementasi fitur ekspor data historis SPL ke dalam format CSV. Data yang diterima dari sistem diubah menjadi struktur baris dan kolom, kemudian disimpan sebagai berkas CSV menggunakan objek Blob. Penambahan karakter Byte Order Mark (BOM) UTF-8 bertujuan menjaga kompatibilitas karakter ketika file dibuka menggunakan aplikasi spreadsheet seperti Microsoft Excel sehingga data historis dapat dianalisis kembali di luar sistem dashboard.

Full Code Aplikasi Web Dashboard

<https://github.com/rohimahmad/Projek-Web-audio-digital-masjid-pintar.git>