

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 Berkat. 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

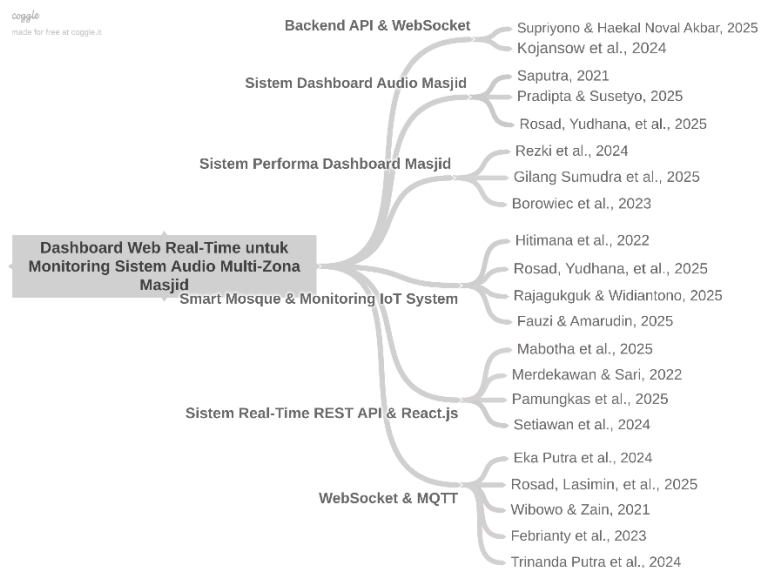
		cuaca dan server <i>room</i> .	berbasis <i>REST API</i> dan <i>WebSocket</i> .	<i>WebSocket</i> untuk pemantauan <i>real-time</i> .
17	Supriyono & Haekal Noval Akbar, 2025	Integrasi <i>REST API</i> + <i>WebSocket</i> .	Fokus <i>backend</i> , tidak ada visualisasi <i>CPS</i> .	<i>Dashboard</i> visualisasi <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</i> admin <i>React</i> & <i>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.

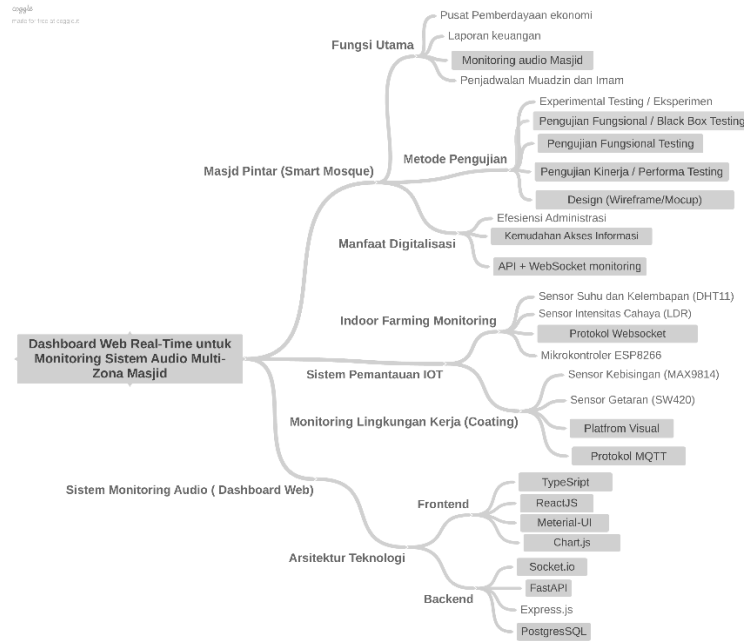
Di sisi lain, perkembangan teknologi *dashboard* berbasis *web* telah memungkinkan penyajian informasi sistem secara lebih interaktif dan mudah dipantau. Berbagai penelitian telah memanfaatkan *framework modern* seperti *ReactJS* serta teknologi komunikasi data *real-time* untuk mendukung kegiatan *monitoring* 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.