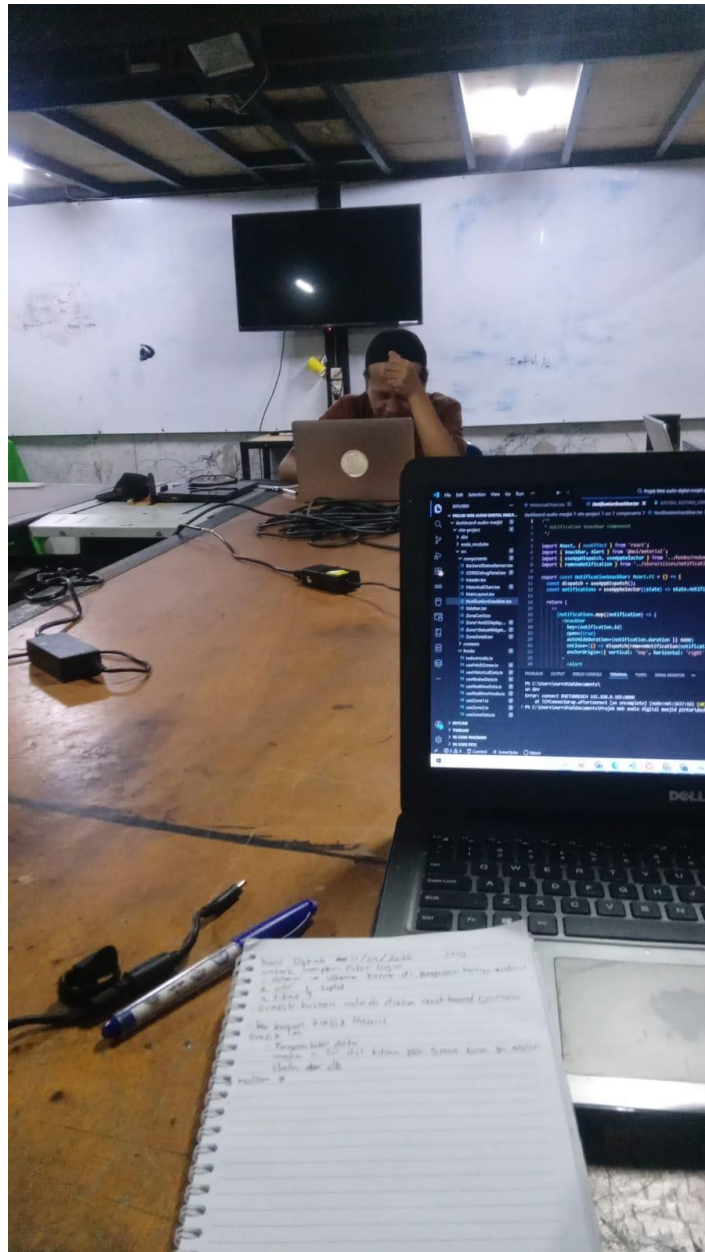


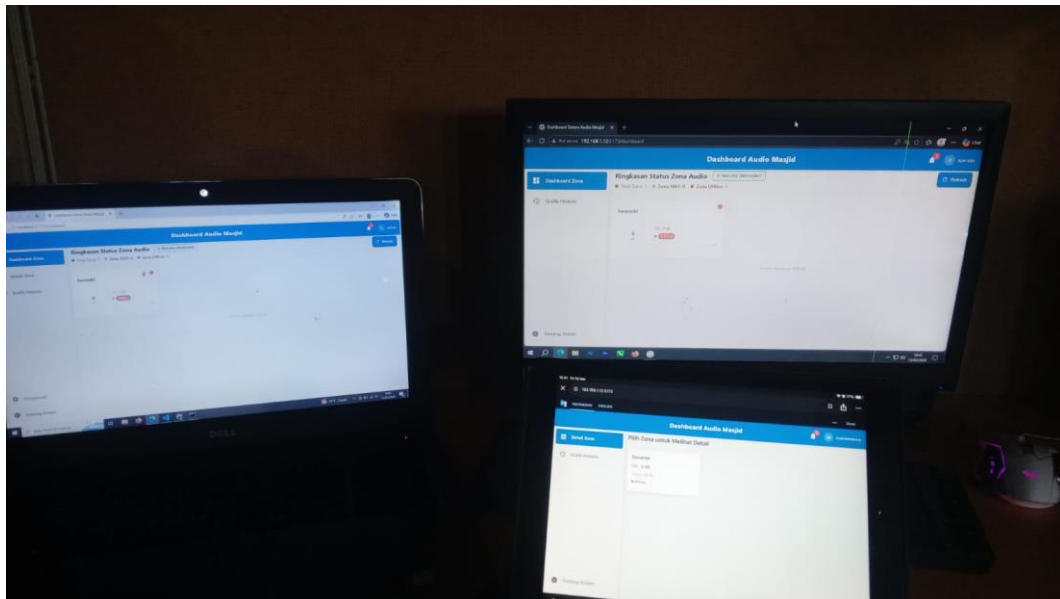
LAMPIRAN

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

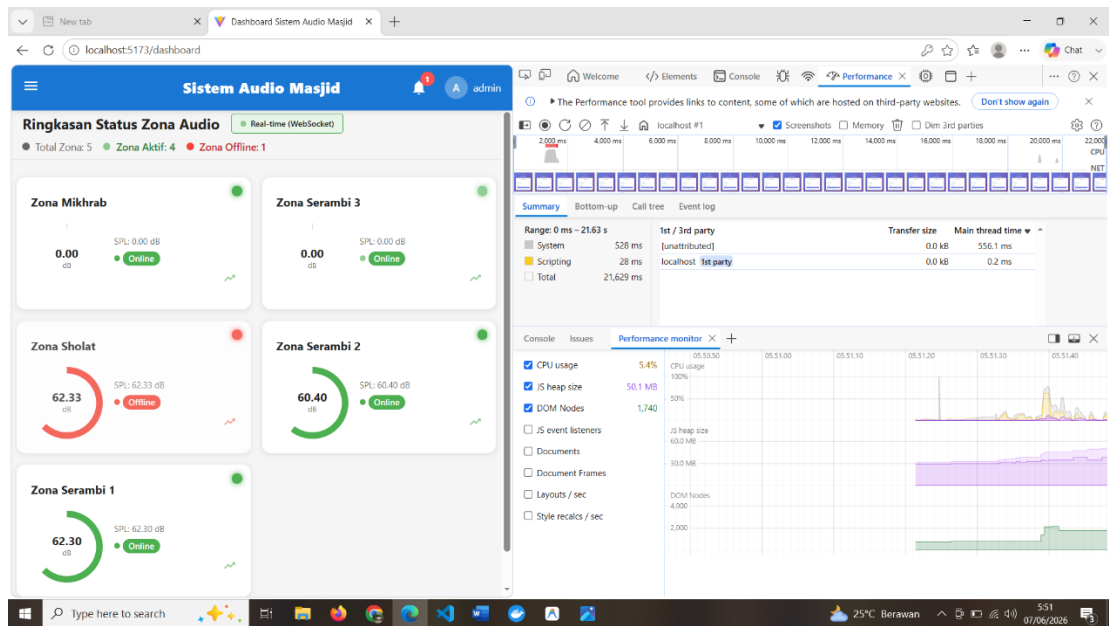
=====	
	HASIL PROFILING DETAIL ZONA (Zona 5)
=====	
Waktu Buka Halaman : 0.87 detik	
Waktu Render Grafik: 0.07 detik (1.440 titik data)	
Status & Metrik : Lengkap (Ya)	
Interaksi Grafik : Smooth (Ya - silakan tes hover)	
Hasil Pengujian : LULUS 	
=====	

Gambar 5. 12 Dokumentasi Pengujian Halaman Detail Zona 5

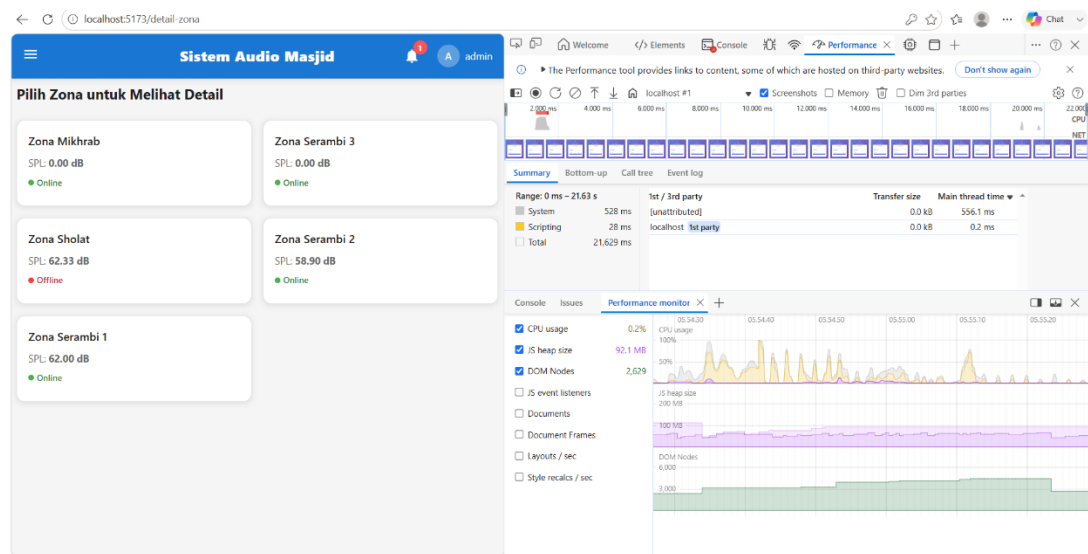
Lampiran 4

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

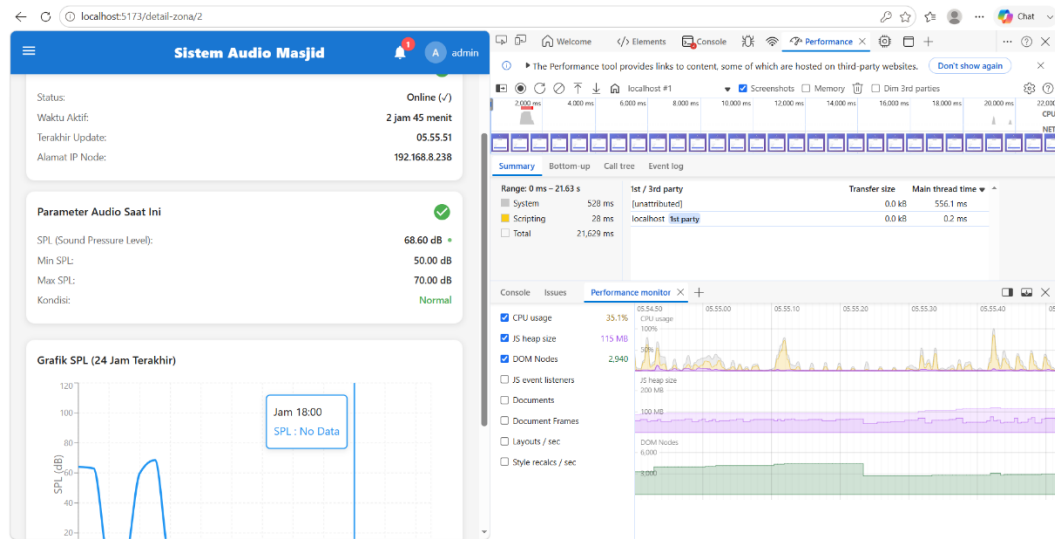
Halaman	FCP (ms)	LCP (ms)	TTI (ms)	Total Load (ms)	Target (ms)	Status
Login	1.240	171,9	1.240	2.114	< 3,0 detik	Lulus
Dashboard overview	1.032	190,8	1982	3.205	< 3,5 detik	Lulus
Detail zona	1.015	130,6	1.383	2.528	< 3,0 detik	Lulus
Grafik historis 1440 points	1.082	849,2	1.925	3.856	< 4,0 detik	Lulus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



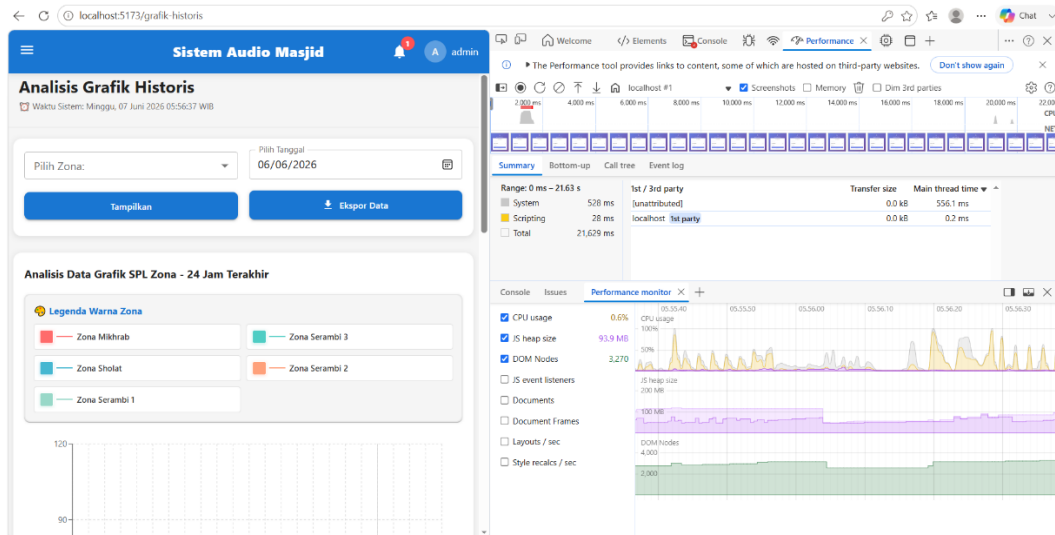
Gambar 5. 13 Dokumentasi Page Load Dashboard Zona



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 telemetri 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 unsubscribe 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.');

```

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>