Panduan Webhook, Realtime Monitoring & Auto Logout

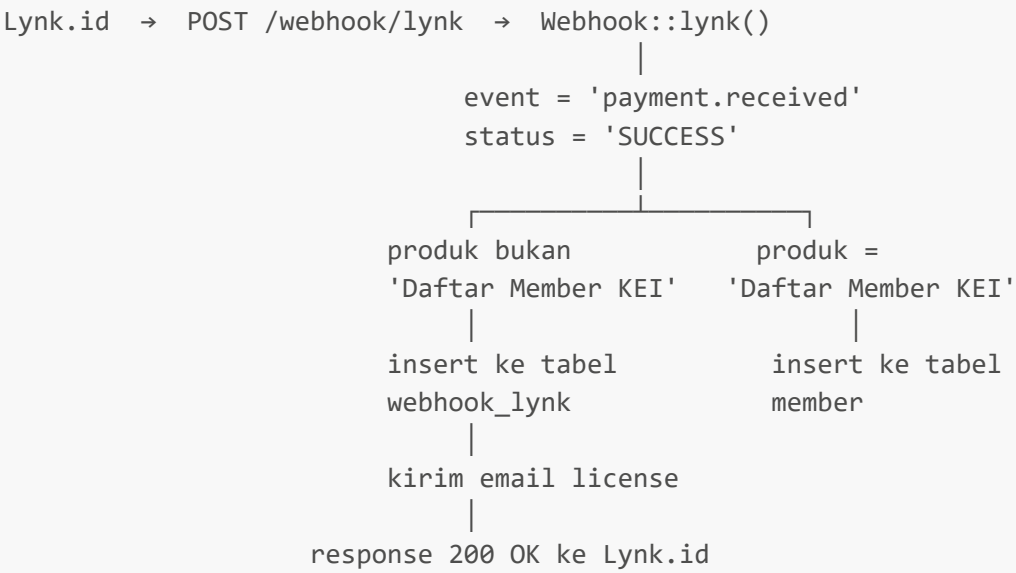
> Dokumen ini dibuat murni berdasarkan kode yang tersedia: `Webhook.php`, `RealtimeDatabaseMonitoring.php`, `Member.php`, `login.php`, `View/member/layout/app.php`, dan `realtime.js`.

Daftar Isi

1. Webhook

Cara Kerja

Lynk.id mengirim `HTTP POST` berisi JSON ke endpoint `/webhook/lynk` setiap kali ada event pembayaran. Controller menerima data itu, lalu memutuskan tindakan berdasarkan isi payload.



Method: `lynk()` — `Webhook.php`

Ambil JSON dari request:

```
$data = $this->request->getJSON(true);
```

Ekstrak field dari payload Lynk.id:

```
$email              = $data['data']['message_data']['customer']['email'];
$nama               = $data['data']['message_data']['customer']['name'];
$no_hp              = $data['data']['message_data']['customer']['phone'];
$event              = $data['event'];
$status             = $data['data']['message_action'];
$transaction_id_lynk = $data['data']['message_data']['refId'];
$time               = $data['data']['message_data']['createdAt'];
$nama_produk        = $data['data']['message_data']['items'][0]['title'];
$price              = $data['data']['message_data']['items'][0]['price'];
$qty                = $data['data']['message_data']['items'][0]['qty'];
$produk_digital     = $data['data']['message_data']['items'][0]['uuid'];
$totals             = $data['data']['message_data']['totals'];
```

Generate ID unik (license, order, transaction):

```
$licenseGenerate = sprintf(
    '%04X-%04X%04X',
    random_int(0, 0xffff),
    random_int(0, 0x0fff) | 0x4000,
    random_int(0, 0x3fff) | 0x8000,
);

$idOrderGenerate = sprintf(
    '%04X%04X',
    random_int(0, 0xffff),
    random_int(0, 0x3fff) | 0x8000,
);

$idTRXGenerate = sprintf(
    '%04X',
    random_int(0, 0x3fff) | 0x8000,
);

$license        = "KEI-" . $licenseGenerate;
$id_order       = "KEI-" . $idOrderGenerate;
$transaction_id = "KEI-" . $idTRXGenerate;
```

Kondisi utama — hanya proses jika event dan status sesuai:

```
if ($event === 'payment.received' && $status === 'SUCCESS') {

    if ($nama_produk !== 'Daftar Member KEI') {
        // Produk digital: simpan license ke tabel webhook_lynk
        $model->insert([
            'email'          => $email,
            'key_license'    => $license,
            'produk_digital' => $produk_digital,
            'id_order'       => $id_order,
```

```
            'id_transaction' => $transaction_id,
            'trx_lynk_id'    => $transaction_id_lynk
        ]);

        // Kirim email berisi license key
        $this->sendPaymentSuccessEmail([
            'order_id'       => $id_order,
            'transaction_id' => $transaction_id,
            'license_key'    => $license,
            'email'          => $email,
            'time'           => $time,
            'nama_produk'    => $nama_produk,
            'jenis'          => 'license'
        ]);

    } else {
        // Pendaftaran member: password = nomor HP yang di-hash
        $hashPassword = password_hash($no_hp, PASSWORD_DEFAULT);

        $member->insert([
            'role'      => 'member',
            'username'  => $email,
            'password'  => $hashPassword,
            'email'     => $email,
            'pic_phone' => $no_hp,
            'pic'       => $nama
        ]);
    }
}

// Wajib selalu response 200 ke Lynk.id
return $this->response
    ->setStatusCode(200)
    ->setJSON(['status' => 'success']);
```

> **Catatan:** Response 200 wajib dikirim selalu — bahkan jika terjadi error di dalam proses — karena jika Lynk.id tidak mendapat 200, mereka akan terus melakukan retry pengiriman webhook.

Model: `WebhookLynk.php` — Tabel `lynk_webhook`

`$model->insert(...)` di atas menggunakan model `WebhookLynk` yang memetakan ke tabel `lynk_webhook`. Berikut kolom-kolom yang diizinkan untuk diisi (`allowedFields`):

```
protected $table         = 'lynk_webhook';
protected $primaryKey    = 'id';
protected $allowedFields = [
    'email',          // email pembeli dari payload Lynk.id
    'key_license',    // license key yang digenerate (format KEI-XXXX-XXXXXXXX)
    'produk_digital', // UUID produk dari payload Lynk.id (items[0]['uuid'])
    'device_active',  // (ada di allowedFields, belum diisi di kode insert saat
ini)
```

```
    'id_order',       // ID order internal (format KEI-XXXXXXXX)
    'id_transaction', // ID transaksi internal (format KEI-XXXX)
    'trx_lynk_id',    // refId transaksi asli dari Lynk.id
];
```

Jadi ketika `$model->insert()` dijalankan, data yang masuk ke baris baru tabel `lynk_webhook` adalah:

Kolom	Sumber Data
`email`	`$data['data']['message_data']['customer']['email']`
`key_license`	Generate lokal — format `KEI-XXXX-XXXXXXXX`
`produk_digital`	`$data['data']['message_data']['items'][0]['uuid']`
`id_order`	Generate lokal — format `KEI-XXXXXXXX`
`id_transaction`	Generate lokal — format `KEI-XXXX`
`trx_lynk_id`	`$data['data']['message_data']['refId']`

> **Catatan:** Kolom `device_active` ada di `allowedFields` tapi tidak diisi saat insert di kode yang ada. Kolom ini kemungkinan disiapkan untuk fitur yang belum diimplementasi.

Method: `sendPaymentSuccessEmail()` — `Webhook.php`

Mengirim email menggunakan email service CodeIgniter. Template email dipilih berdasarkan field `jenis`:

```
private function sendPaymentSuccessEmail($data)
{
    $email = service('email');

    $email->setFrom('no-reply@komunitaseksporindonesia.com', 'Komunitas Ekspor
Indonesia');
    $email->setTo($data['email']);
    $email->setSubject('Pembayaran Berhasil - Order #' . $data['order_id']);

    if ($data['jenis'] === 'license') {
        $message = view('email/email_license', [
            'orderId'       => $data['order_id'] ?? '-',
            'transactionId' => $data['transaction_id'] ?? '-',
            'email'         => $data['email'] ?? '-',
            'license_key'   => $data['license_key'] ?? '-',
            'time'          => $data['time'] ?? '-',
            'name_product'  => $data['nama_produk'] ?? '-',
        ]);
    } else if ($data['jenis'] === 'pendaftaran_member') {
        $message = view('email/email_pendaftaran', [
            'orderId'       => $data['order_id'] ?? '-',
            'transactionId' => $data['transaction_id'] ?? '-',
            'email'         => $data['email'] ?? '-',
            'username'      => $data['username'] ?? '-',
            'time'          => $data['time'] ?? '-',
```

```
            'password'      => $data['password'] ?? '-',
        ]);
    }

    $email->setMessage($message);
    $email->send();
}
```

> 📝 **Catatan:** Email untuk `pendaftaran_member` tidak pernah dipanggil dari kode yang ada saat ini — di blok `else` (Daftar Member KEI), `sendPaymentSuccessEmail` tidak dieksekusi. Kode email pendaftaran ada di method ini tapi belum digunakan.

---

2. Realtime Monitoring (SSE)

Apa itu SSE di Sistem Ini?

Server-Sent Events (SSE) digunakan untuk memantau apakah `fingerprint_device` milik user yang sedang login berubah di database. Perubahan berarti ada login dari perangkat lain — dan browser akan memicu auto logout.

```
Browser buka koneksi ke /api/realtime/attendance-stream
                  ↕ koneksi tetap terbuka
Server cek database setiap 2 detik (sleep(2))
                  |
       fingerprint di DB ≠ fingerprint cookie?
                  |
         Kirim event ke browser
                  |
         Browser → auto logout
```

Method: `attendanceStream()` — `RealtimeDatabaseMonitoring.php`

Konfigurasi wajib sebelum streaming:

```
set_time_limit(0);                          // Matikan batas waktu PHP
ini_set('output_buffering', 'off');         // Matikan buffering output
ini_set('zlib.output_compression', false); // Matikan kompresi
```

Set header SSE:

```
$this->response
    ->setHeader('Content-Type', 'text/event-stream')
    ->setHeader('Cache-Control', 'no-cache')
    ->setHeader('Connection', 'keep-alive')
    ->setHeader('X-Accel-Buffering', 'no'); // untuk Nginx
```

```
$this->response->sendHeaders();

// Bersihkan semua buffer PHP yang aktif
while (ob_get_level() > 0) {
    ob_end_flush();
}
```

Ambil parameter:

```
$fp   = $_COOKIE['device_fp'] ?? null; // fingerprint dari cookie browser
$user = $this->request->getGet('user'); // username dari query string
```

Loop polling selama 25 detik:

```
$start = time();

while (time() - $start < 25) {

    $changes = $this->checkDatabaseChanges($fp, $user);

    if (!empty($changes)) {
        // Kirim data ke browser dalam format SSE
        echo "data: " . json_encode([
            'timestamp' => time(),
            'changes'   => $changes
        ]) . "\n\n";
    } else {
        // Kirim ping agar koneksi tidak terputus
        echo ": ping\n\n";
    }

    flush(); // Paksa PHP kirim output sekarang

    if (connection_aborted()) break; // Hentikan jika browser disconnect

    sleep(2); // Tunggu 2 detik sebelum cek ulang
}

// Setelah 25 detik, minta browser reconnect
echo "event: close\ndata: reconnect\n\n";
flush();
exit; // WAJIB — hentikan eksekusi PHP
```

Method: `checkDatabaseChanges()` — `RealtimeDatabaseMonitoring.php`

Mengecek apakah `fingerprint_device` di database **berbeda** dari yang ada di cookie:

```php
private function checkDatabaseChanges($fp, $user)
{
    $changes = [];

    $updatedAttendances = $this->db->table('member')
        ->where('username', $user)
        ->where('fingerprint_device !=', $fp) // berbeda = ada perangkat lain
        ->limit(1)
        ->get()
        ->getResultArray();

    if (!empty($updatedAttendances)) {
        $changes['updated_attendances'] = $updatedAttendances;
    }

    return $changes;
}
```

Class `RealtimeMonitor` — `realtime.js`

File ini di-load di `View/member/layout/app.php` via `<script src="<?= base_url('js/realtime.js') ?>">`. Di dalamnya terdapat class `RealtimeMonitor` yang dibungkus dalam IIFE dan di-expose ke `window`:

```javascript
(function (global) {
    class RealtimeMonitor {
        constructor(options = {}) {
            this.baseUrl          = options.baseUrl || window.location.origin;
            this.onNewAttendance  = options.onNewAttendance || null;
            this.onUpdateAttendance = options.onUpdateAttendance || null;
            this.onError          = options.onError || null;
            this.onConnected      = options.onConnected || null;
            this.onDisconnected   = options.onDisconnected || null;
            this.user             = options.user || null;

            this.eventSource      = null;
            this.lastCheck        = Math.floor(Date.now() / 1000);
            this.reconnectDelay   = 3000;   // 3 detik
            this.maxReconnectDelay = 30000; // 30 detik
            this.isConnected      = false;
        }

        /**
         * Mulai monitoring
         */
        start() {
            this.connect();
        }

        /**
         * Koneksi ke SSE endpoint
```

```
         */
        connect() {
            const url = `${this.baseUrl}api/realtime/attendance-stream?
user=${this.user}`;

            this.eventSource = new EventSource(url);

            // Event: Koneksi terbuka
            this.eventSource.onopen = () => {
                this.isConnected = true;
                this.reconnectDelay = 3000; // Reset delay
                console.log("[SSE] Connected to server");

                if (this.onConnected) {
                    this.onConnected();
                }
            };

            // Event: Menerima data
            this.eventSource.onmessage = (e) => {
                if (e.data.trim() === "") return; // Skip heartbeat

                try {
                    const data = JSON.parse(e.data);
                    this.lastCheck = data.timestamp;

                    // Handle perubahan
                    this.handleChanges(data.changes);
                } catch (error) {
                    console.error("[SSE] Parse error:", error);
                }
            };

            // Event: Error atau koneksi terputus
            this.eventSource.onerror = (error) => {
                console.error("[SSE] Connection error:", error);
                this.isConnected = false;

                if (this.onDisconnected) {
                    this.onDisconnected();
                }

                // Tutup koneksi dan reconnect
                this.eventSource.close();
                this.reconnect();

                if (this.onError) {
                    this.onError(error);
                }
            };

            // Event: Server minta reconnect
            this.eventSource.addEventListener("close", () => {
                console.log("[SSE] Server requested reconnect");
```

```
            this.eventSource.close();
            this.reconnect();
        });
    }

    /**
     * Reconnect dengan exponential backoff
     */
    reconnect() {
        console.log(`[SSE] Reconnecting in ${this.reconnectDelay}ms...`);

        setTimeout(() => {
            this.connect();
        }, this.reconnectDelay);

        // Tingkatkan delay untuk next reconnect
        this.reconnectDelay = Math.min(
            this.reconnectDelay * 1.5,
            this.maxReconnectDelay,
        );
    }

    /**
     * Handle perubahan dari server
     */
    handleChanges(changes) {
        if (!changes) return;

        // Handle attendance baru
        if (changes.new_attendances && this.onNewAttendance) {
            changes.new_attendances.forEach((attendance) => {
                this.onNewAttendance(attendance);
            });
        }

        // Handle update attendance
        if (changes.updated_attendances && this.onUpdateAttendance) {
            changes.updated_attendances.forEach((attendance) => {
                this.onUpdateAttendance(attendance);
            });
        }
    }

    /**
     * Stop monitoring
     */
    stop() {
        if (this.eventSource) {
            this.eventSource.close();
            this.eventSource = null;
            this.isConnected = false;
            console.log("[SSE] Connection closed");
        }
    }
```

```
        /**
         * Cek status koneksi
         */
        isConnectedToServer() {
            return this.isConnected;
        }
    }

    global.RealtimeMonitor = RealtimeMonitor;
})(window);
```

Method `start()` dan `connect()` — membuka koneksi SSE ke endpoint backend:

```
start() {
    this.connect();
}

connect() {
    // URL endpoint SSE + query param user
    const url = `${this.baseUrl}api/realtime/attendance-stream?user=${this.user}`;

    this.eventSource = new EventSource(url);

    // Koneksi berhasil terbuka
    this.eventSource.onopen = () => {
        this.isConnected = true;
        this.reconnectDelay = 3000; // reset delay setiap koneksi berhasil
        if (this.onConnected) this.onConnected();
    };

    // Terima data dari server
    this.eventSource.onmessage = (e) => {
        if (e.data.trim() === "") return; // skip heartbeat kosong

        try {
            const data = JSON.parse(e.data);
            this.lastCheck = data.timestamp;
            this.handleChanges(data.changes); // proses perubahan
        } catch (error) {
            console.error("[SSE] Parse error:", error);
        }
    };

    // Koneksi error / terputus
    this.eventSource.onerror = (error) => {
        this.isConnected = false;
        if (this.onDisconnected) this.onDisconnected();
        this.eventSource.close();
        this.reconnect(); // coba sambung ulang
        if (this.onError) this.onError(error);
    };
```

```
        // Server kirim event 'close' → minta reconnect
        this.eventSource.addEventListener("close", () => {
            this.eventSource.close();
            this.reconnect();
        });
    }
```

Method `reconnect()` — reconnect dengan exponential backoff:

```
    reconnect() {
        setTimeout(() => {
            this.connect();
        }, this.reconnectDelay);

        // Naikkan delay tiap reconnect gagal, maksimal 30 detik
        this.reconnectDelay = Math.min(
            this.reconnectDelay * 1.5,
            this.maxReconnectDelay,
        );
    }
```

Method `handleChanges()` — memproses data perubahan yang diterima dari server:

```
    handleChanges(changes) {
        if (!changes) return;

        // Jika ada data baru (new_attendances)
        if (changes.new_attendances && this.onNewAttendance) {
            changes.new_attendances.forEach((attendance) => {
                this.onNewAttendance(attendance);
            });
        }

        // Jika ada data yang diupdate (updated_attendances)
        // → ini yang memicu auto logout jika fingerprint berubah
        if (changes.updated_attendances && this.onUpdateAttendance) {
            changes.updated_attendances.forEach((attendance) => {
                this.onUpdateAttendance(attendance);
            });
        }
    }
```

Method `stop()` — menutup koneksi SSE:

```
    stop() {
        if (this.eventSource) {
```

```
            this.eventSource.close();
            this.eventSource = null;
            this.isConnected = false;
        }
    }
```

Inisialisasi di Frontend — `View/member/layout/app.php`

Setelah `realtime.js` di-load, `RealtimeMonitor` diinisialisasi dengan callback yang menentukan reaksi terhadap event dari server:

```
const monitor = new RealtimeMonitor({
    baseUrl: "<?= base_url() ?>",
    user: "<?= esc(session()->get('username')) ?>",

    onNewAttendance: (attendance) => {
        console.log("New attendance:", attendance);
    },

    // Dipanggil ketika checkDatabaseChanges() mendeteksi fingerprint berubah
    onUpdateAttendance: (attendance) => {
        console.log("Updated attendance:", attendance);

        if (attendance.fp !== finger) {
            // Fingerprint di DB berbeda dari cookie → auto logout setelah 2 detik
            setTimeout(() => {
                window.location.href = "/logout";
            }, 2000);
        } else {
            console.log("masih sama");
        }
    },

    onConnected: () => {
        console.log("Realtime monitoring aktif");
    },

    onDisconnected: () => {
        console.log("Koneksi terputus, mencoba reconnect...");
    },

    onError: (error) => {
        console.error("Realtime error:", error);
    },
});
```

Monitoring hanya aktif di halaman `protected-page`:

```
function started() {
    const isProtected = document.body.classList.contains('protected-page');
    const isVideo     = document.body.classList.contains('video-render');

    if (isProtected && !isVideo) {
        monitor.start();
    }
}

document.addEventListener("DOMContentLoaded", () => {
    started();
});

window.addEventListener("beforeunload", () => {
    monitor.stop();
});
```

Penanganan khusus jika halaman memiliki iframe video:

Monitoring ditunda sampai iframe mengirim pesan `VIDEO_READY`, agar tidak konflik saat iframe sedang load:

```
const iframe = document.getElementById('videoFrame');

if (iframe) {
    document.body.classList.add('video-render'); // tunda monitoring

    window.addEventListener('message', e => {
        if (e.data?.type === 'VIDEO_READY') {
            document.body.classList.remove('video-render');
            started(); // baru mulai setelah iframe siap
        }
    });
}
```

---

3. Auto Logout via Device Fingerprint

Konsep

Setiap perangkat/browser memiliki "sidik jari" unik yang dibuat dari properti browser. Sidik jari ini disimpan di cookie dan di database. Jika fingerprint di database berubah (karena ada login dari perangkat lain), perangkat lama di-logout secara otomatis.

Alur Lengkap

```
[Halaman Login]
      |
      ▼
```

```
Generate fingerprint → simpan ke cookie (device_fp)
      │
      ▼
POST /cekfingerprint
  { username, pass, action }
      │
      ├── Password salah / username tidak ada → error 401
      │
      ├── fingerprint_device di DB = null
      │       → update DB dengan fingerprint baru, izinkan login
      │
      ├── fingerprint cookie = fingerprint DB
      │       → izinkan login langsung
      │
      ├── fingerprint BEDA + action = 'keep'
      │       → update DB dengan fingerprint baru, izinkan login
      │
      └── fingerprint BEDA + action != 'keep'
              → valid: false → tampilkan modal konflik

[Modal Konflik]
      ├── "Tetap gunakan di sini" → kirim ulang dengan action='keep'
      └── "Gunakan di perangkat lain" → redirect ke /logout

[Setelah Login — Halaman Protected]
      │
      ▼
SSE aktif → cek fingerprint DB setiap 2 detik
      │
      └── Beda? → auto logout (window.location.href = "/logout")
```

Step 1 — Generate Fingerprint di Browser (`login.php`)

```
function generateFingerprint() {
    const canvas = document.createElement("canvas");
    const ctx = canvas.getContext("2d");
    ctx.textBaseline = "top";
    ctx.font = "14px Arial";
    ctx.fillText("fingerprint", 2, 2);
    const canvasData = canvas.toDataURL();

    const components = [
        navigator.userAgent,
        navigator.language,
        screen.colorDepth,
        screen.width + "x" + screen.height,
        new Date().getTimezoneOffset(),
        !!window.sessionStorage,
        !!window.localStorage,
        canvasData,
    ];
```

```
    return hashString(components.join("|||"));
}

function hashString(str) {
    let hash = 0;
    for (let i = 0; i < str.length; i++) {
        const char = str.charCodeAt(i);
        hash = (hash << 5) - hash + char;
        hash = hash & hash;
    }
    return Math.abs(hash).toString(36);
}

function saveFingerprintToCookie() {
    const fingerprint = generateFingerprint();
    const expires = new Date();
    expires.setFullYear(expires.getFullYear() + 1); // expired 1 tahun
    document.cookie = `device_fp=${fingerprint}; expires=${expires.toUTCString()};
path=/; SameSite=Strict`;
    return fingerprint;
}

function getFingerprint() {
    const match = document.cookie.match(/device_fp=([^;]+)/);
    if (match) return match[1];
    return saveFingerprintToCookie(); // generate baru jika belum ada
}

const fingerprint = getFingerprint();
```

Step 2 — Validasi Fingerprint saat Login (`login.php`)

```
async function fetchHandler(username, pass, action = '') {
    const response = await fetch("/cekfingerprint", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ username, action, pass }),
    });

    const data = await response.json();

    if (data !== '' && data.status !== 'invalid') {
        if (!data.valid) {
            // Fingerprint beda → tampilkan modal konflik
            modal.style.display = 'flex';
            return;
        } else {
            // Lanjut proses login session
            let form = new FormData();
            form.append("username", username);
```

```
            form.append("password", pass);

            const loginResponse = await fetch(data.login, {
                method: "POST",
                body: form
            });

            const responseLogin = await loginResponse.json();

            if (responseLogin.status === 'successful') {
                window.location.href = responseLogin.redirect;
            } else {
                login_message.style.display = 'flex';
                login_message.textContent = responseLogin.message;
            }
        }
    } else if (data !== '' && data.status === 'invalid') {
        login_message.style.display = 'flex';
        login_message.textContent = data.message;
    }
}

// Submit form login
document.getElementById('formLogin').addEventListener('submit', async function(e)
{
    e.preventDefault();
    const username = document.getElementById('username').value;
    const pass     = document.getElementById('password').value;
    fetchHandler(username, pass);
});
```

Login via URL parameter juga didukung:

```
const params = new URLSearchParams(window.location.search);
const hasUsername = params.has('username');
const hasPassword = params.has('password');

if (hasUsername && hasPassword) {
    fetchHandler(params.get('username'), params.get('password'));
}
```

Step 3 — Backend Cek Fingerprint (`Webhook.php`)

```
public function cekFingerprint()
{
    $data     = $this->request->getJSON(true);
    $username = $data['username'];
    $action   = $data['action'];
    $pass     = $data['pass'];
```

```
    $fp = $_COOKIE['device_fp'] ?? null;

    $modelProfil = new Member();
    $fp_database = $modelProfil
        ->where('username', $username)
        ->select('fingerprint_device, password')
        ->first();

    if ($fp_database && password_verify($pass, $fp_database['password'])) {

        // Kasus 1: Fingerprint cocok
        if ($fp === $fp_database['fingerprint_device'] &&
$fp_database['fingerprint_device'] !== null) {
            return response()->setJSON([
                'status' => 'sama', 'valid' => true, 'login' =>
'/auth/authenticate'
            ], 200);
        }

        // Kasus 2: Belum ada fingerprint di DB → simpan fingerprint
        if ($fp_database['fingerprint_device'] === null) {
            $modelProfil->where('username', $username)
                ->set(['fingerprint_device' => $fp, 'action' => 'keep'])
                ->update();
            return response()->setJSON([
                'status' => 'sama', 'valid' => true, 'login' =>
'/auth/authenticate'
            ], 200);
        }

        // Kasus 3: Fingerprint beda tapi user pilih "keep" → update fingerprint
di DB
        if ($fp !== $fp_database['fingerprint_device']
            && $fp_database['fingerprint_device'] !== null
            && $action === 'keep') {
            $modelProfil->where('username', $username)
                ->set(['fingerprint_device' => $fp, 'action' => 'keep'])
                ->update();
            return response()->setJSON([
                'status' => 'sama', 'valid' => true, 'login' =>
'/auth/authenticate'
            ], 200);
        }

        // Kasus 4: Fingerprint beda, belum ada action → kembalikan valid: false
        return response()->setJSON(['status' => 'beda', 'valid' => false], 200);

    } else if (!$fp_database) {
        return response()->setJSON([
            'status' => 'invalid', 'message' => 'Username tidak ada.'
        ], 401);
    } else {
        return response()->setJSON([
```

```
            'status' => 'invalid', 'message' => 'Username atau Password salah.'
        ], 401);
    }
}
```

Step 4 — Modal Konflik Sesi (`login.php`)

Modal ditampilkan ketika server mengembalikan `valid: false`:

```
<div id="alert-modal" class="alert-overlay"> <!-- display: none by default -->
    <div class="alert-modal">
        <div class="alert-body">
            <h3 class="alert-title">Konflik Sesi Terdeteksi</h3>
            <p class="alert-text">
                Website ini (<span
class="domain">komunitaseksporindonesia.com</span>)
                sedang aktif di tab atau perangkat lain.
            </p>
            <div class="info-box">
                Beberapa sesi aktif tidak diizinkan karena alasan keamanan.
            </div>
        </div>

        <div class="alert-footer">
            <button onclick="handleAction('other')" class="btn btn-secondary">
                Gunakan di perangkat lain
            </button>
            <button onclick="handleAction('keep')" class="btn btn-primary">
                Tetap gunakan di sini
            </button>
        </div>
    </div>
</div>
```

```
async function handleAction(action) {
    if (action === 'keep') {
        const username = document.getElementById('username').value;
        const pass     = document.getElementById('password').value;

        if (hasUsername && hasPassword) {
            // Jika login via URL param
            fetchHandler(params.get('username'), params.get('password'), 'keep');
        } else {
            fetchHandler(username, pass, 'keep');
        }
    } else {
        // Pilih "perangkat lain" → logout
        window.location.href = "/logout";
```

```
    }
}
```

Kolom Database yang Terlibat (`Member.php`)

Dua kolom di tabel `member` yang digunakan untuk sistem ini:

```
protected $allowedFields = [
    // ... field lainnya ...
    'fingerprint_device', // menyimpan fingerprint perangkat aktif
    'action'              // menyimpan status 'keep'
];
```

Penutup — Potensi Pengembangan Realtime Monitoring

Sistem realtime monitoring yang dibangun di sini menggunakan SSE untuk memantau perubahan `fingerprint_device` di database sebagai mekanisme keamanan sesi. Namun arsitektur ini tidak terbatas pada kasus tersebut.

Selama memahami alur dasarnya — server polling database di dalam loop SSE, lalu mendorong hasilnya ke browser via `handleChanges()` — sistem ini bisa dikembangkan untuk kebutuhan lain. Salah satu contoh yang relevan adalah memanfaatkan kolom `device_active` yang sudah ada di tabel `lynk_webhook`: setiap kali user membuka produk digital, fingerprint perangkatnya dicatat ke kolom tersebut, lalu SSE memantau apakah ada perangkat lain yang juga mengakses lisensi yang sama. Jika terdeteksi, akses bisa langsung dicabut dari sisi browser tanpa perlu user melakukan refresh — cukup dengan menambahkan query baru di `checkDatabaseChanges()` dan callback yang sesuai di sisi frontend.

Ringkasan File & Fungsi

File	Method / Fungsi	Peran
`Webhook.php`	`lynk()`	Terima & proses payload dari Lynk.id
`Webhook.php`	`sendPaymentSuccessEmail()`	Kirim email setelah pembayaran
`Webhook.php`	`cekFingerprint()`	Validasi fingerprint saat login
`Webhook.php`	`cekAction()`	Cek status action fingerprint
`RealtimeDatabaseMonitoring.php`	`attendanceStream()`	SSE endpoint — stream perubahan DB ke browser
`RealtimeDatabaseMonitoring.php`	`checkDatabaseChanges()`	Query DB: fingerprint berubah atau tidak

File	Method / Fungsi	Peran
`Member.php`	Model	Kolom `fingerprint_device` & `action`
`login.php`	`generateFingerprint()`	Buat fingerprint dari properti browser
`login.php`	`hashString()`	Hash komponen fingerprint ke string base36
`login.php`	`getFingerprint()` / `saveFingerprintToCookie()`	Ambil atau simpan fingerprint ke cookie
`login.php`	`fetchHandler()`	Kirim request ke `/cekfingerprint`
`login.php`	`handleAction()`	Handle tombol modal konflik
`realtime.js`	`constructor()`	Inisialisasi opsi, delay, dan state koneksi
`realtime.js`	`start()` / `connect()`	Buka koneksi EventSource ke SSE endpoint
`realtime.js`	`handleChanges()`	Proses data `new_attendances` & `updated_attendances` dari server
`realtime.js`	`reconnect()`	Reconnect otomatis dengan exponential backoff (maks 30 detik)
`realtime.js`	`stop()`	Tutup koneksi EventSource
`View/member/layout/app.php`	Inisialisasi `RealtimeMonitor`	Set callback `onUpdateAttendance` untuk memicu auto logout