Panduan Sistem Tracking Aktivitas & Rekomendasi Artikel

Overview Alur Sistem

```
User buka artikel
    → (JS) kirim count: true langsung
    → (JS) setiap 10 detik kirim scroll & time
    → (PHP) aktivitas() terima & simpan ke DB
    → (PHP) getRekomendasiByUserActivity() hitung skor
    → tampil rekomendasi di beranda
```

Bagian 1 — Frontend (JavaScript)

Script ini ditempatkan di halaman detail artikel dan bertanggung jawab mengumpulkan serta mengirim data perilaku user ke backend.

1.1 Konfigurasi URL Endpoint

```
const activityUrl = "<?= rtrim(base_url($lang . '/api/v1/aktivitas'), '/') ?>";
```

URL dibuat dinamis menggunakan `base_url()` dari CodeIgniter dengan prefix bahasa (`$lang`), lalu `rtrim` memastikan tidak ada trailing slash di akhir URL.

1.2 Kirim Kunjungan Saat Halaman Dibuka

```
fetch(activityUrl, {
    method: "POST",
    credentials: 'same-origin',
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
        article_kategori: "<?= $artikel['id_kategori'] ?>",
        count: true,   // hanya tambah counter kunjungan
    })
}).catch(err => console.error("Gagal kirim aktivitas:", err));
```

Dikirim **segera saat halaman dimuat** tanpa menunggu DOMContentLoaded. `count: true` memberitahu backend untuk hanya menginkremen `data.count` tanpa memproses time/scroll. `credentials: 'same-origin'` memastikan cookie session ikut terkirim agar `session_id()` di backend terbaca dengan benar.

1.3 Tracking Scroll & Waktu Baca

Inisialisasi Variabel

```
document.addEventListener("DOMContentLoaded", function() {
    let startTime = Date.now(); // waktu mulai membaca
    let maxScroll = 0;          // menyimpan scroll tertinggi yang pernah dicapai
```

`startTime` dicatat sejak DOM siap agar waktu baca lebih akurat.

Kalkulasi Scroll Maksimum

```
window.addEventListener("scroll", function() {
    let scrollTop = window.scrollY;
    let docHeight = document.documentElement.scrollHeight - window.innerHeight;
    let scrollPercent = (scrollTop / docHeight) * 100;
    maxScroll = Math.max(maxScroll, scrollPercent); // simpan nilai tertinggi saja
});
```

`Math.max` memastikan nilai tidak turun ketika user scroll ke atas. Yang dicatat adalah **seberapa jauh paling bawah** user pernah scroll, bukan posisi scroll saat ini.

Fungsi Pembantu `sendActivity()`

```
function sendActivity(type, value) {
    fetch(activityUrl, {
        method: "POST",
        credentials: 'same-origin',
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            article_kategori: "<?= $artikel['id_kategori'] ?>",
            count: false,  // kirim time/scroll, bukan counter kunjungan
            type: type,    // "time_spent_seconds" atau "scroll_percentage"
            value: value
        })
    }).catch(err => console.error("Gagal kirim aktivitas:", err));
}
```

Pengiriman Periodik Setiap 10 Detik

```
setInterval(function() {
    let timeSpent = Math.floor((Date.now() - startTime) / 1000); // ms → detik

    sendActivity("scroll_percentage", maxScroll.toFixed(2));  // contoh: "73.45"
```

```
    sendActivity("time_spent_seconds", timeSpent);          // contoh: 30
}, 10000);
```

Dikirim tiap 10 detik agar backend selalu mendapat data terbaru meski user tidak menutup tab. Nilai `timeSpent` bersifat akumulatif dari awal buka halaman, dan backend akan melakukan **penambahan** (`+=`) ke nilai yang sudah ada di DB.

1.4 Pemetaan Field Frontend → Backend

Field JS	Nilai Contoh	Diproses di Backend
`count: true`	—	`$item['data']['count'] += 1`
`type: "time_spent_seconds"`	`30`	`$item['data']['time'] += 30`
`type: "scroll_percentage"`	`"73.45"`	`$item['data']['scroll'] += 73`
`article_kategori`	`"3"`	dicocokkan dengan `$item['kategori']`

1.5 Catatan Kode yang Dinonaktifkan

`setTimeout` di Pengiriman Awal (Dikomentari)

```
// setTimeout(() => {
//     fetch(activityUrl, { ... count: true })
// }, 9500)
```

Versi sebelumnya menunda pengiriman count selama 9.5 detik agar hanya user yang benar-benar membaca yang terhitung. Dinonaktifkan karena pengiriman langsung lebih simpel, dan filtering "keseriusan membaca" sudah ditangani lewat skor time & scroll di backend.

`beforeunload` dengan `sendBeacon` (Dikomentari)

```
// window.addEventListener("beforeunload", function() {
//     navigator.sendBeacon(activityUrl, timeData);
//     navigator.sendBeacon(activityUrl, scrollData);
// });
```

`sendBeacon` dirancang untuk kirim data saat tab ditutup karena `fetch` biasa bisa terbatalkan. Dinonaktifkan karena `setInterval` sudah mencukupi, dan `sendBeacon` tidak mendukung header `Content-Type: application/json` secara langsung sehingga perlu penanganan tambahan di backend.

Bagian 2 — Backend (PHP / CodeIgniter)

2.1 Fungsi `aktivitas()` — Menerima & Menyimpan Data Tracking

Fungsi ini adalah endpoint API yang menerima POST request dari frontend.

Struktur JSON yang Diterima

```
{
    "article_kategori": "3",
    "type": "time_spent_seconds",
    "value": 45,
    "count": false
}
```

Ambil Session dan Data Tracking yang Sudah Ada

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

$row = $this->MLModel
    ->where('session_id', $sessionId)
    ->select(['id', 'tracking'])
    ->first();

$trackData = json_decode($row['tracking'], true) ?? [];
```

Parsing Nilai Time dan Scroll

Nilai hanya diambil jika `count` bernilai `false`, sehingga request kunjungan (`count: true`) tidak mengubah time/scroll:

```
$intTime   = !$data['count'] ? ($data['type'] === 'time_spent_seconds'  ? (int)
$data['value'] : 0) : 0;
$intScroll = !$data['count'] ? ($data['type'] === 'scroll_percentage'   ? (int)
$data['value'] : 0) : 0;
```

Update Kategori yang Sudah Ada

```
foreach ($trackData as &$item) {
    if ($item['kategori'] === $data['article_kategori']) {

        if (!empty($data['count'])) {
            $item['data']['count'] = ($item['data']['count'] ?? 0) + 1;
        }
```

```
            $item['data']['time']   = ($item['data']['time']   ?? 0) + $intTime;
            $item['data']['scroll'] = ($item['data']['scroll'] ?? 0) + $intScroll;

            $found = true;
            break;
        }
    }
    unset($item); // WAJIB dipanggil setelah foreach dengan reference (&$item)
```

Tambah Entri Baru Jika Kategori Belum Ada

```
if (!$found) {
    $trackData[] = [
        'kategori' => $data['article_kategori'],
        'data'     => ['count' => 1]
    ];
}
```

Simpan Kembali ke DB

```
$this->MLModel->update($row['id'], [
    'tracking' => json_encode($trackData)
]);
```

Contoh Hasil Kolom `tracking` di DB

```
[
    {
        "kategori": "3",
        "data": { "count": 5, "time": 120, "scroll": 75 }
    },
    {
        "kategori": "7",
        "data": { "count": 2, "time": 30, "scroll": 20 }
    }
]
```

2.2 Fungsi `getRekomendasiByUserActivity()` — Kalkulasi Skor & Rekomendasi

Fungsi ini membaca data tracking lalu menghitung skor tiap kategori untuk menentukan artikel yang paling relevan.

Pembagian Kuota Artikel

```
$limitUtama = intdiv($limit, 2) + 1; // limit=5 → limitUtama=3
$limitKedua = $limit - $limitUtama;   // limitKedua=2
```

Artikel dibagi antara dua kategori terbaik agar rekomendasi tidak monoton dari satu kategori saja.

Inisialisasi Session Baru

```
if (!$dataTracking) {
    $this->MLModel->insert(['session_id' => $sessionId]);
}
```

Jika user belum punya data tracking, buat row baru. Fungsi mengembalikan `null` (tidak ada rekomendasi berbasis aktivitas untuk ditampilkan).

Perhitungan Skor Per Kategori

```
$rataRataWaktu  = $time / $count;
$rataRataScroll = $scroll / $count;

// Skor logaritmik — mencegah kategori yang sering dikunjungi mendominasi secara berlebihan
$dataScore[$kategori] = round(($dataScore[$kategori] ?? 0) + log($count + 1), 1);

// Bonus waktu baca: jika rata-rata > 18 detik, user dianggap benar-benar membaca
if ($rataRataWaktu > 18) {
    $dataScore[$kategori] += 3;
} else {
    $dataScore[$kategori] += 1;
}

// Bonus scroll: jika rata-rata > 18%, user scroll cukup jauh ke bawah
if ($rataRataScroll > 18) {
    $dataScore[$kategori] += 3;
}
```

Ringkasan logika skor:

Sinyal	Kondisi	Poin
Kunjungan	Selalu	`log(count + 1)`
Waktu baca	> 18 detik	+3
Waktu baca	≤ 18 detik	+1
Scroll	> 18%	+3

Ambil Artikel Berdasarkan Skor Tertinggi

```
arsort($dataScore); // urutkan dari skor tertinggi
$bestLabel = array_slice($dataScore, 0, 2, true); // ambil 2 kategori terbaik
$labels    = array_keys($bestLabel);

if (count($bestLabel) > 1) {
    $dataUtama  = $this->artikelModel->where('id_kategori', $labels[0])-
>findAll($limitUtama);
    $dataKedua  = $this->artikelModel->where('id_kategori', $labels[1])-
>findAll($limitKedua);
    $dataRekomendasi = array_merge($dataUtama, $dataKedua);
} else {
    $dataRekomendasi = $this->artikelModel->where('id_kategori', $labels[0])-
>findAll($limit);
}

return $this->addCategoryInfo($dataRekomendasi);
```

2.3 Fungsi Pendukung `addCategoryInfo()`

Menambahkan data objek kategori ke tiap artikel agar view bisa langsung mengaksesnya tanpa query tambahan:

```
protected function addCategoryInfo(array $articles)
{
    foreach ($articles as &$article) {
        if (!isset($article['kategori'])) {
            $article['kategori'] = $this->kategoriModel-
>find($article['id_kategori']);
        }
    }
    return $articles;
}
```

Bagian 3 — Skema Database

```
CREATE TABLE ml_tracking (
    id          INT PRIMARY KEY AUTO_INCREMENT,
    session_id  VARCHAR(100) UNIQUE NOT NULL,
    tracking    LONGTEXT DEFAULT NULL
    -- Format kolom tracking (JSON array):
    -- [
    --   { "kategori": "3", "data": { "count": 5, "time": 120, "scroll": 75 } },
    --   { "kategori": "7", "data": { "count": 2, "time": 30,  "scroll": 20 } }
```

```
    -- ]
);
```

Bagian 4 — Tips Pengembangan Lanjutan

Frontend

- **Debounce scroll listener** — event `scroll` terpanggil sangat sering. Tambahkan throttle tiap 200ms agar tidak memberatkan performa halaman:

```
let lastScrollTime = 0;
window.addEventListener("scroll", function() {
    const now = Date.now();
    if (now - lastScrollTime < 200) return;
    lastScrollTime = now;
    // ... hitung scrollPercent
});
```

- **Visibility API** — pause `startTime` ketika user pindah tab agar waktu baca yang terekam benar-benar waktu aktif:

```
document.addEventListener("visibilitychange", function() {
    if (document.hidden) {
        pausedAt = Date.now();
    } else {
        startTime += (Date.now() - pausedAt); // kompensasi waktu tab tidak aktif
    }
});
```

- **Deduplikasi tab** — jika user membuka artikel sama di dua tab, `session_id` akan sama dan count bisa dobel. Gunakan `sessionStorage` sebagai guard:

```
const articleKey = `tracked_<?= $artikel['id_artikel'] ?>`;
if (!sessionStorage.getItem(articleKey)) {
    sessionStorage.setItem(articleKey, '1');
    // kirim count: true
}
```

Backend

- **Decay faktor** — beri bobot lebih rendah pada aktivitas lama dengan menyimpan timestamp per kategori, agar skor tidak stagnan untuk user lama.
- **Hindari artikel yang sudah dibaca** — simpan array `id_artikel` yang sudah dikunjungi di kolom tracking, lalu gunakan `whereNotIn()` saat query rekomendasi.

- **Batasi ukuran tracking** — jika `trackData` terlalu banyak entry, trim kategori dengan skor terendah agar payload JSON tidak membengkak seiring waktu.