📅 Breakdown Harian - Minggu 10: Payment Gateway Integration

🎯 Tujuan Minggu Ini

- Memahami konsep dasar Payment Gateway (Midtrans/Xendit).
- Mampu mengintegrasikan frontend React dengan backend untuk proses transaksi pembayaran.
- Memahami cara backend meng-handle notifikasi pembayaran (webhook).
- Mampu mengupdate status pesanan berdasarkan notifikasi pembayaran.

📅 Hari 1 (Senin) - Konsep Payment Gateway & Setup Sandbox

Materi Inti (2 Jam)

- Pengenalan Payment Gateway: Apa itu, mengapa penting, dan bagaimana cara kerjanya.
- Perkenalan Midtrans/Xendit: Fitur utama, jenis pembayaran yang didukung.
- Konsep Sandbox/Development Environment: Pentingnya menggunakan mode sandbox untuk testing.
- Proses registrasi akun sandbox dan mendapatkan API Key (Client Key, Server Key).
- Memahami alur pembayaran dasar (redirect vs. pop-up).

Praktik Mandiri (8 Jam)

- Registrasi akun sandbox di Midtrans atau Xendit.
- Dapatkan API Key (Client Key dan Server Key) untuk mode sandbox.
- Baca dokumentasi dasar Midtrans/Xendit untuk memahami cara membuat transaksi.
- Buat file konfigurasi di proyek backend Anda untuk menyimpan API Key (gunakan environment variables).
- Lakukan percobaan sederhana untuk memanggil API Payment Gateway (misal: cek status API) dari backend menggunakan `curl` atau Postman.

📅 Hari 2 (Selasa) - Integrasi Frontend (React) & Backend untuk Transaksi

Materi Inti (2 Jam)

- Alur integrasi pembayaran: Frontend meminta token transaksi dari backend, backend berkomunikasi dengan Payment Gateway.
- Membuat endpoint API di backend untuk membuat transaksi pembayaran (misal: `/api/payments/create-transaction`).
- Data yang dibutuhkan untuk membuat transaksi (jumlah, detail produk, info user).
- Mengirim permintaan dari frontend (React) ke backend untuk memulai transaksi.
- Menerima response dari backend (token transaksi/redirect URL) dan mengarahkan user ke halaman pembayaran Payment Gateway.

Praktik Mandiri (8 Jam)

- Di backend, buat endpoint POST `/api/payments/create-transaction`.
- Endpoint ini akan menerima data keranjang/pesanan dari frontend.
- Gunakan library resmi Midtrans/Xendit (misal: `midtrans-client` atau `xendit-node`) untuk membuat transaksi baru.
- Kirimkan `transaction_details` dan `item_details` ke Payment Gateway.
- Kembalikan `transaction_token` (Midtrans) atau `invoice_url` (Xendit) ke frontend.
- Di frontend React, buat fungsi `handlePayment` yang memanggil endpoint backend ini setelah proses checkout.
- Setelah mendapatkan token/URL, arahkan user ke halaman pembayaran Payment Gateway (misal: `snap.pay(token)` untuk Midtrans atau `window.location.href = invoice_url` untuk Xendit).

📆 Hari 3 (Rabu) - Backend Meng-handle Notifikasi Pembayaran (Webhook)

Materi Inti (2 Jam)

- Konsep Webhook/Notification URL: Bagaimana Payment Gateway memberitahu backend tentang status pembayaran.
- Pentingnya validasi signature/hash dari notifikasi untuk keamanan.
- Membuat endpoint API di backend untuk menerima notifikasi (misal: `/api/payments/notification`).
- Memparsing data notifikasi dan memahami status pembayaran (success, pending, failed).

Praktik Mandiri (8 Jam)

- Di backend, buat endpoint POST `/api/payments/notification`.
- Konfigurasi Notification URL ini di dashboard sandbox Payment Gateway Anda.
- Implementasikan logika untuk memvalidasi notifikasi (cek signature key/hash).
- Parsing data JSON yang diterima dari notifikasi.
- Log status pembayaran yang diterima (misal: `transaction_status`, `fraud_status`).
- Lakukan simulasi pembayaran di sandbox dan pastikan notifikasi diterima oleh endpoint backend Anda.

📆 Hari 4 (Kamis) - Update Status Pesanan

Materi Inti (2 Jam)

- Pentingnya mengupdate status pesanan di database setelah pembayaran berhasil.
- Logika untuk menangani berbagai status pembayaran (pending, success, expire, cancel).
- Menggunakan ID transaksi dari Payment Gateway untuk mencocokkan dengan pesanan di database Anda.
- Penanganan kasus edge: notifikasi duplikat, notifikasi yang terlambat.

Praktik Mandiri (8 Jam)

- Di database Anda, tambahkan kolom `payment_status` (misal: 'pending', 'paid', 'failed', 'expired') dan `payment_gateway_transaction_id` di tabel `orders` atau `transactions`.

- Di endpoint notifikasi backend (`/api/payments/notification`), setelah validasi, update status pesanan di database Anda berdasarkan `transaction_status` dari notifikasi.
- Pastikan untuk mencocokkan pesanan menggunakan `order_id` atau `transaction_id` yang Anda kirimkan ke Payment Gateway.
- Implementasikan logika untuk mengupdate stok produk jika pembayaran berhasil (opsional).
- Uji coba berbagai skenario pembayaran (sukses, pending, gagal) di sandbox dan verifikasi status pesanan di database Anda.

📆 Hari 5 (Jum'at) - Review & Tugas Mingguan

Materi Inti (2 Jam)

- Review menyeluruh alur Payment Gateway dari frontend hingga backend dan update status.
- Membahas potensi masalah dan solusi (misal: idempotency, retry mechanism).
- Persiapan Tugas Mingguan.

Tugas Mingguan (Waktu Pengerjaan: Sisa Minggu + Akhir Pekan)

Tema Tugas: Implementasi Integrasi Payment Gateway di Aplikasi E-commerce.

Persyaratan Fungsional:

1. **Integrasi Payment Gateway:** Implementasikan integrasi dengan Midtrans atau Xendit (mode sandbox).
2. **Endpoint Transaksi Backend:** Buat endpoint di backend untuk membuat transaksi pembayaran dan berkomunikasi dengan Payment Gateway.
3. **Frontend Payment Flow:** Di frontend React, implementasikan alur untuk memulai pembayaran setelah checkout, mengarahkan user ke halaman pembayaran Payment Gateway.
4. **Webhook Notifikasi:** Buat endpoint di backend untuk menerima dan memvalidasi notifikasi status pembayaran dari Payment Gateway.
5. **Update Status Pesanan:** Berdasarkan notifikasi, update status pesanan di database Anda (misal: dari 'pending' menjadi 'paid').
6. **Dokumentasi (`README.md`):** Perbarui file `README.md` dengan penjelasan alur Payment Gateway dan cara mengujinya.

Fitur Opsional (Nilai Tambah):

- Implementasi halaman sukses/gagal pembayaran di frontend.
- Penanganan error yang lebih robust saat komunikasi dengan Payment Gateway.
- Pencatatan log notifikasi pembayaran untuk debugging.
- Implementasi fitur refund (membutuhkan API Payment Gateway dan logika backend).

Output Tugas:

- Kode sumber aplikasi React dan backend yang sudah terintegrasi Payment Gateway.
- File `README.md` yang diperbarui.
- Push semua perubahan ke repositori GitHub pribadi Anda.

🗣️ Sesi Presentasi (Waktu disesuaikan)

Siapkan diri Anda untuk sesi presentasi singkat di awal minggu berikutnya. Fokus presentasi adalah:

- **Demo:** Tunjukkan alur checkout hingga proses pembayaran (simulasi di sandbox) dan bagaimana status pesanan berubah.
- **Penjelasan Implementasi:** Jelaskan bagaimana frontend dan backend berkomunikasi dengan Payment Gateway, serta bagaimana notifikasi ditangani.
- **Tantangan & Solusi:** Ceritakan tantangan yang Anda hadapi saat mengerjakan tugas dan bagaimana Anda menyelesaikannya.

💡 Tips Belajar Tambahan

- **Keamanan:** Selalu pastikan API Key disimpan dengan aman (environment variables) dan validasi notifikasi webhook.
- **Idempotency:** Pertimbangkan bagaimana menangani notifikasi duplikat untuk mencegah update status yang tidak diinginkan.
- **Logging:** Aktifkan logging di backend untuk memantau request dan response dari Payment Gateway serta notifikasi.

Minggu ini kita akan masuk ke bagian krusial dari aplikasi e-commerce: pembayaran. Integrasi Payment Gateway melibatkan komunikasi yang kompleks antara frontend, backend, dan layanan pihak ketiga. Fokus pada pemahaman alur data dan penanganan status pembayaran yang akurat. Selamat belajar dan mengerjakan tugas!