Penjelasan midtrans

Midtrans merupakan sistem pembayaran yang dapat memfasilitasi penjual dan pembeli untuk melakukan transaksi. Midtrans menyediakan tools terintegrasi ke e-commerce sesuai kebutuhan pembayaran secara online dengan kartu debit, kartu kredit, bahkan penarikan uang, dan pengiriman uang.

Pertama .env diatur untuk menghubungkan wesite dengan midtrans seperti erikut

```
MIDTRANS_MERCHANT_ID=G335780542
MIDTRANS_CLIENT_KEY=SB-Mid-client-w24VozsjJ0foDdfp
MIDTRANS_SERVER_KEY=SB-Mid-server-aUKhZxiNR3eh7Bf3o6MUw2BB
MIDTRANS_PRODUCTION=false
MIDTRANS_SANITIZED=true
MIDTRANS_3DS=true
```

Diisi sesuai dengan yang ada di akun midtran

Membuat function untuk menerima status pembayaran dari midtrans

```
public function webhooks(Request $request)
    {
        $auth = base64_encode(env('MIDTRANS_SERVER_KEY'));

        $response = Http::withHeaders([
            'Content-Type' => 'application/json',
            'Authorization' => "Basic $auth",
        ])->get("https://api.sandbox.midtrans.com/v2/$request-
>order_id/status");


        $response = json_decode($response->body());

        $tagihan = TagihanKerusakan::where('id', $request->order_id)->first();

        if ($tagihan->status == 'settlement' || $tagihan->status == 'capture')
{
            return response()->json('Payment has been already processed');
        }

        if ($response->transaction_status == 'capture') {
            $tagihan->status = 'capture';
        } elseif ($response->transaction_status == 'settlement') {
            $tagihan->status = 'settlement';
        } elseif ($response->transaction_status == 'pending') {
            $tagihan->status = 'pending';
        } elseif ($response->transaction_status == 'deny') {
            $tagihan->status = 'deny';
        } elseif ($response->transaction_status == 'cancel') {
            $tagihan->status = 'cancel';
```

```
        } elseif ($response->transaction_status == 'expire') {
            $tagihan->status = 'expire';
        } elseif ($response->transaction_status == 'refund') {
            $tagihan->status = 'refund';
        } else {
            $tagihan->status = 'error';
        }
        $tagihan->save();

        return response()->json(['status' => 'success']);
    }
```

Method `webhooks` yang Anda berikan berfungsi untuk menerima notifikasi dari Midtrans melalui webhook, yang mengindikasikan status pembayaran yang terkait dengan tagihan. Webhook ini adalah cara Midtrans memberi tahu aplikasi Anda tentang status transaksi pembayaran, seperti "settlement", "capture", "deny", dan lainnya. Berikut penjelasan langkah-langkah dari kode tersebut:

1. Mendapatkan Authorization

```
$auth = base64_encode(env('MIDTRANS_SERVER_KEY'));
```

- Kode ini mengonversi **`MIDTRANS_SERVER_KEY`** yang disimpan di file `.env` menjadi format **Base64** untuk otentikasi saat mengakses API Midtrans. Ini digunakan untuk membuat header otorisasi yang diperlukan untuk validasi API Midtrans.

2. Meminta Status Pembayaran dari Midtrans

```
$response = Http::withHeaders([
    'Content-Type' => 'application/json',
    'Authorization' => "Basic $auth",
])->get("https://api.sandbox.midtrans.com/v2/$request->order_id/status");
```

- Menggunakan HTTP client Laravel (`Http::withHeaders`) untuk mengirim permintaan **GET** ke API Midtrans untuk mendapatkan status pembayaran transaksi. URL API yang digunakan adalah `https://api.sandbox.midtrans.com/v2/{order_id}/status`, di mana `{order_id}` digantikan dengan ID tagihan yang diterima dari webhook (`$request->order_id`).
- Header **Authorization** berisi server key Midtrans yang sudah dienkode menjadi format base64.
- API Midtrans akan mengembalikan status transaksi untuk `order_id` yang dimaksud.

3. Mengonversi Response JSON

```
$response = json_decode($response->body());
```

- Respons yang diterima dari Midtrans diubah menjadi format objek PHP menggunakan **`json_decode`** agar bisa diakses dengan mudah.

4. Mencari Tagihan Berdasarkan `order_id`

```
$tagihan = TagihanKerusakan::where('id', $request->order_id)->first();
```

- Mencari tagihan dalam tabel `TagihanKerusakan` berdasarkan `order_id` yang dikirimkan dalam request. Ini memungkinkan untuk memperbarui status tagihan yang sesuai dengan hasil pembayaran.

5. Cek Status Pembayaran dan Pembaruan Tagihan

```
if ($tagihan->status == 'settlement' || $tagihan->status == 'capture') {
    return response()->json('Payment has been already processed');
}
```

- Jika tagihan sudah memiliki status `'settlement'` atau `'capture'`, yang berarti pembayaran sudah diproses sebelumnya, maka tidak ada pembaruan status yang perlu dilakukan, dan sistem mengembalikan pesan yang mengatakan bahwa pembayaran sudah diproses.

6. Memperbarui Status Berdasarkan `transaction_status`

- Status transaksi yang diterima dari Midtrans (`$response->transaction_status`) akan digunakan untuk memperbarui status tagihan. Berikut adalah kondisi yang digunakan:
  - **`capture`**: Pembayaran berhasil diproses dan dana telah dicatat.
  - **`settlement`**: Pembayaran telah selesai dan diterima.
  - **`pending`**: Pembayaran sedang menunggu.
  - **`deny`**: Pembayaran ditolak.
  - **`cancel`**: Pembayaran dibatalkan.
  - **`expire`**: Pembayaran telah kedaluwarsa.
  - **`refund`**: Pembayaran telah dikembalikan.
  - Jika status transaksi tidak cocok dengan salah satu dari kondisi di atas, status tagihan akan diubah menjadi `'error'`.

Setiap kondisi ini akan memperbarui status tagihan yang sesuai di database:

```
if ($response->transaction_status == 'capture') {
    $tagihan->status = 'capture';
} elseif ($response->transaction_status == 'settlement') {
    $tagihan->status = 'settlement';
} elseif ($response->transaction_status == 'pending') {
    $tagihan->status = 'pending';
} elseif ($response->transaction_status == 'deny') {
    $tagihan->status = 'deny';
} elseif ($response->transaction_status == 'cancel') {
    $tagihan->status = 'cancel';
} elseif ($response->transaction_status == 'expire') {
    $tagihan->status = 'expire';
} elseif ($response->transaction_status == 'refund') {
    $tagihan->status = 'refund';
} else {
    $tagihan->status = 'error';
}
```

7. Menyimpan Status yang Diperbarui

```
$tagihan->save();
```

- Setelah status tagihan diperbarui, perubahan tersebut disimpan ke dalam database.

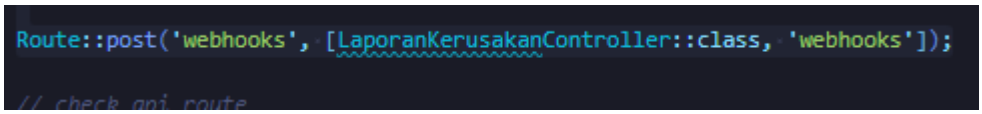
8. Mengirim Respons JSON

```
return response()->json(['status' => 'success']);
```

- Setelah webhook diproses dan status diperbarui, server mengembalikan respons JSON yang berisi status `'success'` untuk menandakan bahwa webhook telah diterima dan diproses dengan sukses.

Method `webhooks` ini menangani notifikasi dari Midtrans yang memberi tahu tentang status pembayaran. Jika status transaksi berubah, misalnya menjadi "settlement" atau "capture", maka status tagihan dalam database akan diperbarui sesuai dengan status terbaru dari Midtrans. Fungsi ini juga memeriksa apakah pembayaran sudah diproses sebelumnya untuk menghindari pembaruan status yang tidak perlu.

Membuat routing untuk method terseut



Setelah itu mengaktifdkan ngrok sebagai tunneling agar localhost kita dapat diakses midtrans

Dengan cmd : ngrok http 8000 (sesuaikan dengan port masing masing)

Setelah itu link ngrok tadi ditambahkan dengan routing ini, agar hasil dari midtrans dapat terkirim ke website

Ambil link pada ngrok simpan dalam konfigurasi midtrans pada platform midtrans

Membuat function untuk membuat tagihan, di **LaporanKerusakanController**

createTagihan($id)

```
public function createTagihan($id)
{
    $laporan_kerusakan = LaporanKerusakan::findOrFail($id);
    $detail_peminjaman = DetailPeminjaman::where('id', $laporan_kerusakan-
>id_detail_peminjaman)->first();
    $inventaris = Inventaris::where('id', $detail_peminjaman->id_inventaris)-
>first();
    return view('laporan.kerusakan.create_tagihan',
compact('laporan_kerusakan', 'detail_peminjaman', 'inventaris'));
}
```

- Fungsi ini digunakan untuk menampilkan halaman pembuatan tagihan.

- **$laporan_kerusakan = LaporanKerusakan::findOrFail($id)**: Mencari laporan kerusakan berdasarkan ID yang diberikan. Jika tidak ditemukan, akan menghasilkan error.
- **$detail_peminjaman = DetailPeminjaman::where('id', $laporan_kerusakan->id_detail_peminjaman)->first()**: Mendapatkan detail peminjaman berdasarkan ID yang ada pada laporan kerusakan.
- **$inventaris = Inventaris::where('id', $detail_peminjaman->id_inventaris)->first()**: Mendapatkan informasi inventaris terkait dengan detail peminjaman.
- **return view('laporan.kerusakan.create_tagihan', compact('laporan_kerusakan', 'detail_peminjaman', 'inventaris'))**: Mengirimkan data yang didapatkan ke view untuk ditampilkan.

2. storeTagihan(Request $request)

```
public function storeTagihan(Request $request)
    {
        $request->validate(
            [
                'id_lk' => 'required|exists:laporan_kerusakan,id',
                'biaya_perbaikan' => 'required|numeric',
            ],
            [
                'id_lk.required' => 'ID Laporan Kerusakan harus diisi.',
                'id_lk.exists' => 'ID Laporan Kerusakan tidak ditemukan.',
                'biaya_perbaikan.required' => 'Biaya perbaikan harus diisi.',
                'biaya_perbaikan.numeric' => 'Biaya perbaikan harus berupa
angka.',
            ]
        );

        $idLK = $request->id_lk;
        $total_harga = $request->biaya_perbaikan;
        $modelTagihan = new TagihanKerusakan();
        if ($idLK = $modelTagihan->where('id_laporan_kerusakan', $idLK)-
>first()) {
            return redirect()->back()->with('error', 'Tagihan sudah dibuat.');
        }
        $dataInput = ([
            'id_laporan_kerusakan' => $request->id_lk,
            'total_tagihan' => $total_harga,
            'status' => 'pending',
        ]);
        // dd($data);
        $modelTagihan->create($dataInput);
        $idTagihan = $modelTagihan->latest()->first()->id;
        $model_laporan = new LaporanKerusakan();
        $data = $model_laporan->getPeminjaman($request->id_lk);
        $params = [
```

```
            'transaction_details' => [
                'order_id' => $idTagihan,
                'gross_amount' => $total_harga,
            ],
            'customer_details' => [
                'first_name' => $data->name,
                'email' => $data->email,
                'phone' => $data->phone,
            ],
        ];
        $paymentUrl = Snap::createTransaction($params)->redirect_url;
        $snapToken = $snapToken = Snap::getSnapToken($params);
        // dd($paymentUrl, $params);

        $modelTagihan->where('id', $idTagihan)->update([
            'token' => $snapToken,
            'payment_url' => $paymentUrl,
        ]);

        $this->sendEmailPenagihan($idTagihan);

        return redirect()->route('laporan_kerusakan.index')->with('success',
'Tagihan berhasil dibuat.');
    }
```

- Fungsi ini digunakan untuk menyimpan tagihan yang dibuat oleh pengguna.

Validasi Input

- **$request->validate([...])**: Melakukan validasi terhadap input yang diberikan oleh pengguna, yaitu:
    - id_lk: ID Laporan Kerusakan harus diisi dan ada di dalam tabel laporan_kerusakan.
    - biaya_perbaikan: Biaya perbaikan harus diisi dan berupa angka.

Cek Tagihan yang Sudah Dibuat

- **if ($idLK = $modelTagihan->where('id_laporan_kerusakan', $idLK)->first())**: Mengecek apakah sudah ada tagihan terkait dengan laporan kerusakan yang diberikan. Jika sudah ada, sistem akan mengembalikan pesan error dan tidak melanjutkan pembuatan tagihan baru.

Membuat Tagihan Baru

- **$dataInput = [...]**: Menyiapkan data yang akan dimasukkan ke dalam tabel tagihan_kerusakan. Data yang dimasukkan antara lain:
    - id_laporan_kerusakan: ID Laporan Kerusakan.
    - total_tagihan: Biaya perbaikan yang diberikan oleh pengguna.
    - status: Status tagihan yang diatur menjadi 'pending'.

- **$modelTagihan->create($dataInput)**: Membuat tagihan baru berdasarkan data yang disiapkan.
- **$idTagihan = $modelTagihan->latest()->first()->id**: Mengambil ID tagihan terbaru yang baru saja dibuat.

Mengintegrasikan dengan Midtrans

- **$model_laporan->getPeminjaman($request->id_lk)**: Mengambil data peminjam berdasarkan ID laporan kerusakan.
- **$params = [...]**: Menyusun parameter untuk transaksi pembayaran menggunakan Midtrans, yang mencakup:
    - transaction_details: Menentukan ID tagihan dan jumlah total tagihan (biaya perbaikan).
    - customer_details: Menyediakan informasi pelanggan (nama, email, dan nomor telepon).
- **$paymentUrl = Snap::createTransaction($params)->redirect_url**: Menggunakan Midtrans API untuk membuat transaksi dan mendapatkan URL pembayaran yang akan diarahkan ke pelanggan.
- **$snapToken = Snap::getSnapToken($params)**: Mendapatkan token Snap untuk transaksi Midtrans, yang diperlukan untuk menyelesaikan pembayaran.

Update Tagihan dengan URL dan Token Pembayaran

- **$modelTagihan->where('id', $idTagihan)->update([...])**: Mengupdate tagihan dengan token Snap dan URL pembayaran yang didapat dari Midtrans.

Mengirim Email Penagihan

- **$this->sendEmailPenagihan($idTagihan)**: Mengirimkan email penagihan kepada pelanggan setelah tagihan berhasil dibuat. Ini bisa berisi detail tagihan dan link untuk melakukan pembayaran.

Menyelesaikan Proses

- **return redirect()->route('laporan_kerusakan.index')->with('success', 'Tagihan berhasil dibuat.')**: Setelah semua proses selesai, pengguna akan diarahkan ke halaman daftar laporan kerusakan dengan pesan sukses.

Setelah itu memuat function/method untuk mengirimkan link pebayaran kepada peminjam

```php
public function sendEmailPenagihan($id)
    {
        $tagihanModel = new TagihanKerusakan();
        $tagihan = $tagihanModel->getLaporanPeminjaman($id);
        // dd($tagihan);
        Mail::to($tagihan->email)->send(new TagihanPenggantian($tagihan));

        return response()->json(['status' => 'Email sent successfully']);
    }
```

Method `sendEmailPenagihan($id)` berfungsi untuk mengirimkan email penagihan kepada pelanggan terkait tagihan yang telah dibuat, khususnya untuk pembayaran atas kerusakan atau penggantian barang. Berikut adalah penjelasan mendetail tentang setiap langkah dalam kode tersebut:

```
$tagihanModel = new TagihanKerusakan();
        $tagihan = $tagihanModel->getLaporanPeminjaman($id);
```

- Baris pertama membuat instance baru dari model **`TagihanKerusakan`** yang digunakan untuk berinteraksi dengan tabel `tagihan_kerusakan` di database.
- Baris kedua memanggil fungsi **`getLaporanPeminjaman($id)`** untuk mengambil data tagihan berdasarkan ID yang diberikan. Fungsi ini kemungkinan mengembalikan data tagihan yang lengkap, termasuk informasi terkait pelanggan (seperti email), status tagihan, dan informasi lainnya.
- **`$tagihan`** akan berisi objek yang menyimpan informasi tentang tagihan yang akan digunakan untuk mengirimkan email.

```
Mail::to($tagihan->email)->send(new TagihanPenggantian($tagihan));
```

- **`Mail::to($tagihan->email)`**: Menggunakan facade **`Mail`** dari Laravel untuk mengirimkan email. Fungsi **`to()`** menentukan alamat email penerima, yang diambil dari properti **`email`** dalam objek `$tagihan`.
- **`send(new TagihanPenggantian($tagihan))`**: Fungsi **`send()`** mengirimkan email menggunakan kelas email **`TagihanPenggantian`**, yang merupakan kelas Mailable di Laravel.

  - **`TagihanPenggantian`** adalah kelas Mailable yang bertanggung jawab untuk mendefinisikan bagaimana email tersebut dibentuk (seperti subjek, isi email, dan template).
  - Objek `$tagihan` yang diteruskan ke kelas **`TagihanPenggantian`** memungkinkan kelas ini untuk mengakses data tagihan dan menyertakan informasi tersebut dalam email.
  - `return response()->json(['status' => 'Email sent successfully']);`

Setelah email dikirim, method ini mengembalikan respons JSON yang berisi status `'Email sent successfully'`. Ini mengindikasikan bahwa proses pengiriman email berhasil dilakukan.