Laporan

QRIS Payment Mindtrans

Dosen Pengampu : Saiful Nur Budiman, S. Kom., M.Kom

Disusun Oleh :

Nafisa Ahlam Gisyeilla (23104410082)

Santi Kurniawati (23104410059)

Tia Neiska Puspita (23104410081)

Desi Widyawati (23104410090)

Annisa Kusuma Dewi (23104410100)

PRODI TEKNIK INFORMATIKA

FAKULTAS TEKNI DAN INFORMATIKA

UNIVERSITAS ISLAM BALITAR

Oktober 2025

KATA PENGANTAR

Puji syukur diucapkan kehadirat Allah Swt. Atas segala rahmat-Nya sehingga makalah ini dapat tersusun sampai selesai. Tidak lupa penulis mengucapkan terima kasih terhadap bantuan dari pihak yang telah berkontribusi dengan memberikan sumbangan baik pikiran maupun materi.

Penulis sangat berharap semoga laporan ini dapat menambah pengetahuan pembaca. Bahkan penulis berharap lebih jauh lagi agar laporan ini dapat membuat pembaca lebih paham akan pembahasan yang dibahas di dalam makalah ini.

Penulis merasa bahwa masih banyak kekurangan dalam penyusunan laporan ini karena keterbatasan pengetahuan dan pengalama. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan laporan ini.

Blitar, 13 Oktober 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi mendorong meningkatnya kebutuhan akan sistem transaksi digital yang cepat, aman, dan mudah diintegrasikan ke berbagai aplikasi. Salah satu bentuk penerapannya adalah penggunaan payment gateway, yaitu layanan yang menjadi penghubung antara pengguna, aplikasi, dan penyedia layanan keuangan. Dalam dunia pengembangan web, framework Laravel banyak digunakan karena memiliki struktur yang rapi dan mendukung integrasi API pihak ketiga, termasuk sistem pembayaran seperti Midtrans.

Midtrans sendiri merupakan layanan payment gateway lokal Indonesia yang menyediakan berbagai metode pembayaran seperti kartu kredit, transfer bank, e-wallet, dan QRIS. Agar pengujian sistem pembayaran ini bisa dilakukan di lingkungan lokal tanpa harus diunggah ke server publik, digunakan Ngrok, sebuah alat yang membuat tunnel sementara agar sistem lokal bisa diakses dari internet. Untuk mendukung tampilan antarmuka yang modern dan responsif, digunakan juga Tailwind CSS sebagai framework desain yang efisien.

Integrasi Laravel, Midtrans, dan Ngrok memungkinkan pengembang membangun dan menguji sistem pembayaran QRIS secara end-to-end, mulai dari pembuatan transaksi, tampilan halaman pembayaran, hingga penerimaan notifikasi status pembayaran. Oleh karena itu, penelitian ini membahas implementasi Payment Gateway Integration menggunakan Laravel, Midtrans, dan Ngrok sebagai langkah nyata dalam memahami cara kerja API pembayaran dan penerapannya dalam aplikasi web modern.

1.2 Rumusan Masalah

1. Bagaimana proses integrasi sistem pembayaran Midtrans ke dalam aplikasi berbasis Laravel?
2. Bagaimana konfigurasi dan pengujian sistem pembayaran dapat dilakukan secara lokal menggunakan Ngrok?
3. Bagaimana implementasi tampilan antarmuka pembayaran dapat dibuat menggunakan Tailwind CSS agar lebih responsif?
4. Bagaimana cara sistem memverifikasi status transaksi setelah pembayaran dilakukan melalui API Midtrans?

1.3 Tujuan Penulisan

1. Menjelaskan langkah-langkah integrasi Midtrans ke dalam aplikasi Laravel sebagai payment gateway.
2. Menunjukkan cara penggunaan Ngrok untuk menguji webhook atau callback Midtrans pada server lokal.
3. Mengimplementasikan tampilan sistem pembayaran berbasis Laravel dengan Tailwind CSS.
4. Menguji dan menganalisis hasil implementasi sistem pembayaran digital menggunakan QRIS melalui Midtrans.

BAB II

LANDASAN TEORI

2.1 Laravel

Laravel adalah framework PHP yang mengikuti pola arsitektur MVC (Model-View-Controller). Laravel menyediakan routing, ORM (Eloquent), Blade templating, queue, middleware, serta tooling frontend (Vite) yang mempercepat pengembangan web modern. Untuk integrasi payment gateway, struktur controller, service class, dan routing di Laravel memudahkan pemisahan logika server-side (membuat order, memanggil API Midtrans, menerima callback). Instalasi dan setup Laravel dapat dilakukan via Composer dan dokumentasi resmi Laravel menguraikan praktik instalasi, environment (.env), serta valet/valet/ngrok untuk pengembangan lokal.

2.2 Midtrans

Midtrans menyediakan API (Snap & Core API) yang memungkinkan merchant membuat transaksi dan menampilkan halaman pembayaran yang sudah dioptimalkan. Dua kunci penting yang digunakan adalah Server Key (dipakai di backend untuk panggilan API yang sensitif) dan Client Key (dipakai di frontend untuk fitur tertentu seperti Snap JS). Alur integrasi umum: server menghasilkan token/transaction payload → frontend memanggil Midtrans Snap untuk membuka widget pembayaran → user menyelesaikan pembayaran → Midtrans mengirim notifikasi (webhook/callback) ke server merchant untuk update status transaksi. Dokumentasi Midtrans menjelaskan detail endpoint, autentikasi, dan cara menangani callback.

2.3 Ngrok

Ngrok adalah layanan tunneling yang membuat URL publik sementara yang meneruskan request ke server lokal. Ini berguna untuk menguji webhook dari layanan eksternal (seperti Midtrans) tanpa perlu deploy ke server publik. Laravel Valet memudahkan integrasi ngrok untuk pengguna macos; namun ngrok juga dapat digunakan secara mandiri di berbagai OS. Saat menggunakan ngrok untuk webhook, developer harus memastikan URL callback terdaftar di dashboard Midtrans (atau menyesuaikan callback pada environment sandbox) dan memperhatikan aspek keamanan (mis. Verifikasi signature).

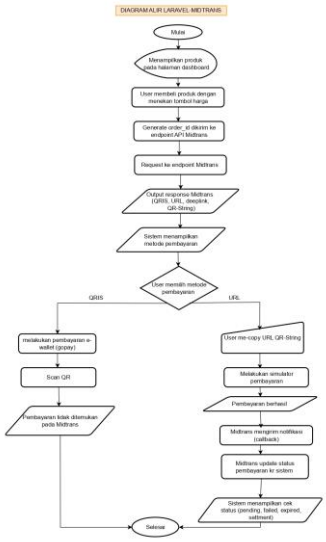
2.4 Tailwind CSS

Tailwind adalah framework utility-first CSS yang menonjolkan kelas-kelas utilitas untuk styling langsung di markup, mempercepat frontend prototyping. Dalam konteks Laravel, Tailwind biasanya diintegrasikan lewat Vite/NPM sehingga developer dapat dengan cepat membangun UI pembayaran yang responsif (butuh konfigurasi build pipeline: node, npm/yarn/bun, dan vite).

BAB III

PEMBAHASAN

3.1 Diagram Alir



Penjelasanya :

1. Mulai : Proses dimulai ketika pengguna (user) membuka aplikasi atau website vending machine.

2. Menampilkan produk di halaman dashboard : Sistem menampilkan daftar produk yang tersedia di dashboard — misalnya makanan, minuman, atau item digital — lengkap dengan harga masing-masing.

3. User membeli produk dengan menekan tombol harga : Ketika user memilih produk dengan menekan tombol harga, sistem mulai memproses transaksi pembayaran.

4. Generate order_id dan kirim ke endpoint API Midtrans : Sistem Laravel membuat ID pesanan unik (order_id) dan mengirimkannya ke API Midtrans untuk memulai transaksi.

5. Request ke endpoint Midtrans untuk pembayaran e-wallet (GoPay) : Laravel mengirim permintaan (request) ke server Midtrans agar pembayaran bisa dilakukan menggunakan metode e-wallet seperti GoPay.

6. Output response dari Midtrans

Midtrans mengirimkan response yang berisi beberapa data penting: QRIS, URL pembayaran, Deeplink, QR String.

Data ini digunakan agar user bisa melakukan pembayaran dengan mudah.

7. Sistem menampilkan metode pembayaran : Website Laravel menampilkan opsi pembayaran berdasarkan data yang diterima dari Midtrans (misalnya QRIS, GoPay, dll).

8. User memilih metode pembayaran : User memilih salah satu metode (misalnya QRIS).

9. Scan QR / Copy URL QR String

User bisa melakukan pembayaran dengan cara: Memindai kode QR yang tampil, atau Menyalin (copy) URL atau QR String dan membuka di aplikasi e-wallet.

10. Simulator pembayaran : Pada tahap pengujian (sandbox), sistem melakukan simulasi pembayaran untuk meniru proses transaksi nyata.

11. Pembayaran berhasil : Jika proses berhasil, Midtrans menganggap transaksi sukses.

12. Midtrans mengirim notifikasi (callback) : Midtrans mengirimkan callback notification ke endpoint Laravel (biasanya /api/payment/notification) untuk memberi tahu status terbaru transaksi.

13. Sistem menampilkan status pembayaran

Laravel menampilkan hasil akhir seperti: Pending, Failed, Expired, Settlement (berhasil).

14. Midtrans update status pembayaran ke sistem : Midtrans memperbarui status transaksi di database aplikasi agar sinkron dengan status sebenarnya di server Midtrans.

15. Pembayaran tidak ditemukan (opsional) : Jika Midtrans tidak menemukan pembayaran (misalnya user tidak melakukan scan QR), sistem menampilkan pesan error atau status gagal.

16. Selesai : Proses transaksi selesai, baik sukses maupun gagal.

3.2 Source Code

1. Paymentcontroller.php

a. Namespace dan Import

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Midtrans\Config;
```

Bagian ini mendefinisikan lokasi file di dalam struktur proyek Laravel (App\Http\Controllers) dan memanggil class penting: Request untuk menerima data dari pengguna, Log untuk mencatat aktivitas atau error, serta Config dari Midtrans untuk pengaturan kunci server dan mode pembayaran.

b. Konfigurasi Midtrans

```php
public function __construct()
{
    Config::$serverKey = config(key: 'midtrans.server_key');
    Config::$isProduction = config(key: 'midtrans.is_production', default: false);
    Config::$isSanitized = true;
    Config::$is3ds = true;
}
```

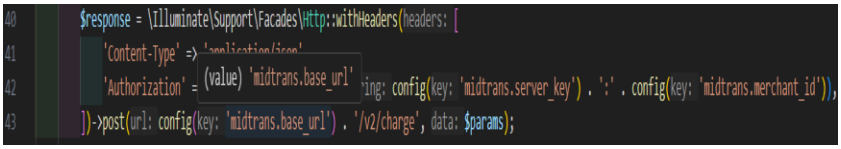
Bagian ini dijalankan otomatis saat controller dibuat. Fungsinya untuk mengatur konfigurasi dasar Midtrans, seperti server key, mode (sandbox atau production), serta keamanan transaksi. Dengan konfigurasi ini, sistem dapat terhubung ke API Midtrans dengan benar.

c. Membuat Transaksi Midtrans

```php
public function payment(Request $request): View
{
    $product_id = $request->query(key: 'product_id', default: 1);
    $price = $request->query(key: 'price', default: 10000);
    $orderId = 'ORDER-' . rand(min: 1000, max: 9999);

    $params = [
        'payment_type' => 'qris',
        'transaction_details' => [
            'order_id' => $orderId,
            'gross_amount' => $price,
        ],
        'qris' => [
            'acquirer' => 'gopay',
        ],
    ];
```

Fungsi ini membuat transaksi baru berdasarkan data dari request pengguna. Ia menetapkan ID produk, harga, dan membuat nomor order unik. Parameter transaksi dikirim ke Midtrans dengan tipe pembayaran QRIS (gopay). Bagian ini juga menggunakan *try-catch* untuk menangani error dan menyimpan log request/response ke file log Laravel.

d. Proses Permintaan ke API Midtrans

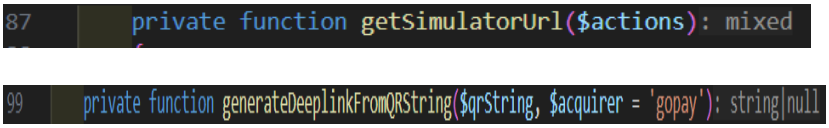


Kode ini mengirim permintaan HTTP POST ke endpoint Midtrans (/v2/charge) dengan header berisi autentikasi menggunakan *server key*. Hasilnya adalah respon JSON berisi data pembayaran, termasuk URL QRIS dan deeplink yang digunakan untuk menampilkan kode pembayaran ke pengguna.

e. Pengolahan Hasil dan Tampilan View

Setelah mendapatkan respon dari Midtrans, data seperti QR code URL dan deeplink dikirim ke view bernama payment.blade.php. View ini menampilkan tampilan pembayaran kepada pengguna agar mereka bisa memindai QRIS atau membuka aplikasi e-wallet (gopay, OVO, dll.) Untuk membayar.

f. Fungsi Bantu Simulator dan Generate Deep Link



Fungsi ini memeriksa status transaksi berdasarkan order_id. Ia mengirim permintaan GET ke API Midtrans untuk mengetahui apakah pembayaran sudah sukses, pending, atau gagal. Hasilnya dikembalikan dalam format JSON agar bisa ditampilkan di halaman status pembayaran atau API response.

g. Mengecek Status Pembayaran

Fungsi ini memeriksa status transaksi berdasarkan order_id. Ia mengirim permintaan GET ke API Midtrans untuk mengetahui apakah pembayaran sudah sukses, pending, atau gagal. Hasilnya dikembalikan dalam format JSON agar bisa ditampilkan di halaman status pembayaran atau API response.

h. Fungsi Index (Halaman Produk)

```php
    public function index(): View
    {
        $products = config(key: 'products.list');
        return view(view: 'prepayment', data: compact(var_name: 'products'));
    }
}
```

Fungsi ini menampilkan halaman awal berisi daftar produk sebelum pengguna melakukan pembayaran. Data produk diambil dari file konfigurasi config/products.php dan dikirim ke view prepayment.blade.php.

2. Prepayment.blade.php

a. Bagian Header

```html
<!DOCTYPE html>
<html lang="en" class="h-full bg-gray-300">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  {{--  --}}
  @vite(entrypoints: 'resources/css/app.css')
  {{-- mengatur profile dropdown --}}
  <script defer src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"></script>
  <title>Vending Machine</title>
</head>
```

Bagian ini berisi struktur dasar HTML yang mengatur tampilan agar responsif dan kompatibel di berbagai perangkat. @vite('resources/css/app.css') memuat gaya dari Tailwind CSS, sedangkan Alpine.js digunakan untuk fitur interaktif seperti dropdown. Judul halaman ditetapkan sebagai “Vending Machine.”

b. Navbar

```html
<!-- Navbar -->
<nav class="bg-gray-800/70 backdrop-blur-md border-b border-gray-700">
  <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div class="flex h-16 items-center justify-between">
      <!-- Left side -->
      <div class="flex items-center space-x-4">
        <img src="https://tailwindcss.com/plus-assets/img/logos/mark.svg?color=indigo&shade=500" alt="Logo" class="size-8">
        <span class="font-semibold text-white text-lg">Vending Machine</span>
      </div>/.flex.items-center.space-x-4

      <!-- Right side -->
      <div class="flex items-center space-x-6">
        <!-- Nav links -->
        <a href="/prepayment" class="text-gray-300 hover:text-white text-sm font-medium">...</a>
        <a href="/prepayment" class="text-gray-300 hover:text-white text-sm font-medium">...</a>
        <a href="/prepayment" class="text-gray-300 hover:text-white text-sm font-medium">...</a>

```

```html
        <!-- Profile dropdown -->
        <div x-data="{ open: false }" class="relative">
          <button @click="open = !open" class="flex items-center rounded-full focus:outline-none focus:ring-2 focus:ring-indigo-500">
            <img src="https://t4.ftcdn.net/jpg/11/66/06/77/360_F_1166067709_2SooAuPWXp20XkGev7oOT7nuK1VThCsN.jpg" class="size-8 rounded-full" alt="Avatar">
          </button>/.flex.items-center.rounded-full.focus:outline-none.focus:ring-2.focus:ring-indigo-500

          <!-- Dropdown -->
          <div x-show="open" @click.outside="open = false"
            x-transition
            class="absolute right-0 z-20 mt-2 w-48 rounded-md bg-gray-800 shadow-lg ring-1 ring-white/10">
            <a href="/prepayment" class="block px-4 py-2 text-sm text-gray-300 hover:bg-white/5 hover:text-white">#</a>
            <a href="/prepayment" class="block px-4 py-2 text-sm text-gray-300 hover:bg-white/5 hover:text-white">#</a>
            <a href="/prepayment" class="block px-4 py-2 text-sm text-gray-300 hover:bg-white/5 hover:text-white">#</a>
          </div>/.absolute.right-0.z-20.mt-2.w-48.rounded-md.bg-gray-800.shadow-lg.ring-1.ring-white/10
        </div>/.relative
      </div>/.flex.items-center.space-x-6
    </div>/.flex.h-16.items-center.justify-between
  </div>/.max-w-7xl.mx-auto.px-4.sm:px-6.lg:px-8
</nav>/.bg-gray-800/70.backdrop-blur-md.border-b.border-gray-700
```

Navbar menampilkan logo dan nama aplikasi di kiri serta menu profil di kanan. Alpine.js digunakan untuk membuat dropdown profil interaktif, dan Tailwind memberikan efek transparan dan buram sehingga tampil modern dan rapi.

c. Header Konten

```
<!-- Header -->
<header class="bg-gray-800 border-y border-gray-700">
  <div class="max-w-7xl mx-auto px-4 py-6 sm:px-6 lg:px-8">
    <h1 class="text-3xl font-bold text-white">All Products</h1>
  </div>/.max-w-7xl.mx-auto.px-4.py-6.sm:px-6.lg:px-8
</header>/.bg-gray-800.border-y.border-gray-700
```

Bagian header ini berfungsi untuk menampilkan judul halaman “All Products” di atas daftar produk. Elemen ini memiliki latar belakang berwarna abu gelap (bg-gray-800) dan garis batas atas bawah (border-y border-gray-700), membuat judul tampak menonjol serta menjadi pembatas antara navbar dan isi utama halaman.

d. Maincontent

```
<!-- Product Card -->
@foreach ($products as $product)
  <div class="bg-gray-400 rounded-xl shadow p-4 hover:scale-110 transition-transform">
    <img src="{{ $product['url'] }}"
    alt="Snack" class="w-80 h-40 object-cover rounded-md">/.w-80.h-40.object-cover.rounded-md
    <h2 class="mt-3 text-base font-semibold text-white">{{ $product['name'] }}</h2>
      <a href="{{ route(name: 'payment', parameters: ['product_id' => $product['id'], 'price' => $product['price']]) }}">
        <button class="mt-3 w-full bg-indigo-600 hover:bg-indigo-900 text-white font-medium py-2 rounded-lg">
          Rp{{ number_format(num: $product['price']) }}
        </button>/.mt-3.w-full.bg-indigo-600.hover:bg-indigo-900.text-white.font-medium.py-2.rounded-lg
      </a>
  </div>/.bg-gray-400.rounded-xl.shadow.p-4.hover:scale-110.transition-transform
@endforeach

</div>/.grid.grid-cols-1.sm:grid-cols-2.md:grid-cols-3.lg:grid-cols-4.gap-6-
</main>/.max-w-7xl.mx-auto.px-4.py-10.sm:px-6.lg:px-8
```

Bagian ini menampilkan daftar produk secara dinamis menggunakan @foreach, di mana setiap produk ditampilkan dalam kartu berisi gambar, nama, dan tombol harga yang terhubung ke route pembayaran. Tampilan diatur dengan Tailwind CSS agar responsif dan memiliki efek interaktif saat di-hover.

e. Footer

```
<footer class="mt-16 text-center text-gray-500 text-sm pb-6">
  &copy; {{ date(format: 'Y') }} Vending Machine | All rights reserved.
</footer>/.mt-16.text-center.text-gray-500.text-sm.pb-6

</body>/.h-full.text-gray-200
</html>
```

Bagian footer menampilkan teks hak cipta dengan tahun otomatis ({{ date('Y') }}) menggunakan sintaks Blade Laravel. Teks diletakkan di tengah (text-center) dengan warna abu muda (text-gray-500) dan ukuran kecil (text-sm), memberikan penutup halaman yang sederhana namun informatif.

3. Product.php

```
<?php

return [
    // List of products
    'list' => [
        [
            'id'    => 'P000001',
            'name'  => 'Sosro',
            'url'   => 'https://www.static-src.com/wcsstore/Indraprastha/images/catalog/full//98/MTA-3541787/sosro_sosro-teh-botol-kotak--330-ml-_full02.jpg',
            'price' => 10000,
        ],
```

Kode di atas adalah file konfigurasi PHP yang berisi daftar produk dalam bentuk array. Setiap produk memiliki atribut id, name, url, dan price untuk menyimpan kode unik, nama, gambar, dan harga produk. File ini berfungsi sebagai data katalog sementara yang digunakan dalam aplikasi Laravel untuk menampilkan produk di halaman toko atau proses pembayaran.

4. Payment.blade.php

a. Bagian struktur awal dan resource

```
<!DOCTYPE html>
<html lang="en" class="h-full bg-gray-300">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Payment | Vending Machine</title>
  @vite(entrypoints: 'resources/css/app.css')
  {{-- AlpineJS untuk interaktivitas kecil --}}
  <script defer src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"></script>
```

Bagian ini merupakan struktur awal dokumen HTML yang berfungsi untuk mendefinisikan jenis dokumen (<!DOCTYPE html>), bahasa (lang="en"), serta pengaturan tampilan penuh halaman dengan latar belakang abu-abu muda (bg-gray-300). Tag <head> memuat metadata seperti pengaturan karakter (UTF-8), viewport untuk responsivitas, dan judul halaman “Payment | Vending Machine”. Selain itu, terdapat integrasi Tailwind CSS menggunakan @vite (fitur Laravel untuk mengompilasi aset modern) dan alpinejs yang digunakan untuk interaktivitas ringan di sisi frontend tanpa perlu framework besar seperti Vue atau React.

b. Bagian navbar

```
<nav class="bg-gray-800/70 backdrop-blur-md border-b border-gray-700">
  <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div class="flex h-16 items-center justify-between">
      <div class="flex items-center space-x-4">
        <img src="https://tailwindcss.com/plus-assets/img/logos/mark.svg?color=indigo&shade=500"
             alt="Logo" class="size-8">/.size-8
        <span class="font-semibold text-white text-lg">Vending Machine</span>
      </div>/.flex.items-center.space-x-4
    </div>/.flex.h-16.items-center.justify-between
  </div>/.max-w-7xl.mx-auto.px-4.sm:px-6.lg:px-8
</nav>/.bg-gray-800/70.backdrop-blur-md.border-b.border-gray-700
```

Bagian ini adalah navigasi utama (navbar) yang menampilkan logo dan nama aplikasi “Vending Machine”. Menggunakan class Tailwind seperti bg-gray-800/70 untuk memberi efek transparan gelap dan backdrop-blur-md agar tampak modern. Struktur flex digunakan agar elemen logo dan teks sejajar secara horizontal serta responsif di berbagai ukuran layar. Elemen ini menjadi bagian identitas visual aplikasi.

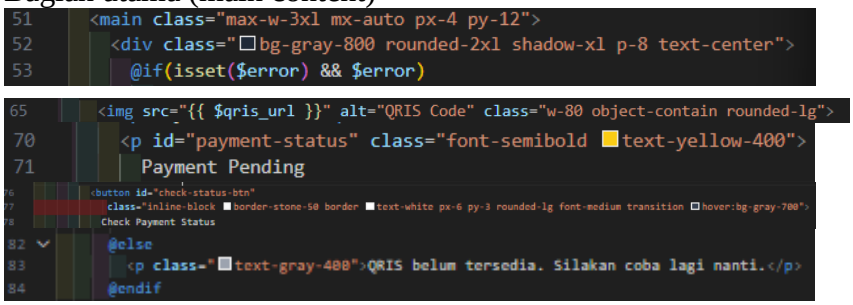
c. Header halaman

```
<header class="bg-gray-800 border-y border-gray-700">
  <div class="max-w-7xl mx-auto px-4 py-6 text-center">
    <h1 class="text-3xl font-bold text-white">MAKE A PAYMENT</h1>
    <p class="text-gray-400 mt-2">Please make your payment by scanning the QRIS code below</p>
  </div>/.max-w-7xl.mx-auto.px-4.py-6.text-center
</header>/.bg-gray-800.border-y.border-gray-700
```

Bagian header berfungsi sebagai judul utama halaman pembayaran, memberikan instruksi kepada pengguna untuk melakukan pembayaran melalui QRIS. Menggunakan desain sederhana dan terpusat (text-center) dengan warna kontras

antara teks putih dan latar belakang abu gelap, menjadikan tampilan profesional dan mudah dibaca oleh pengguna.

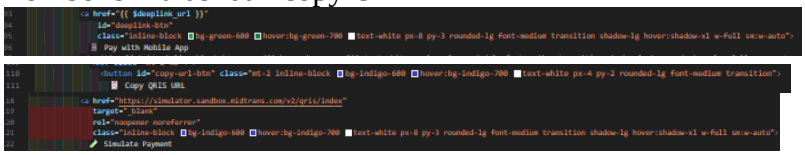
d. Bagian utama (main content)



```
<main class="max-w-3xl mx-auto px-4 py-12">
  <div class="bg-gray-800 rounded-2xl shadow-xl p-8 text-center">
    @if(isset($error) && $error)
    <img src="{{ $qris_url }}" alt="QRIS Code" class="w-80 object-contain rounded-lg">
      <p id="payment-status" class="font-semibold text-yellow-400">
        Payment Pending
      <button id="check-status-btn"
        class="inline-block border-stone-50 border text-white px-6 py-3 rounded-lg font-medium transition hover:bg-gray-700">
        Check Payment Status
      @else
        <p class="text-gray-400">QRIS belum tersedia. Silakan coba lagi nanti.</p>
      @endif
```

Bagian utama ini merupakan inti dari tampilan pembayaran. Ia menampilkan QRIS yang diambil dari variabel {{ $qris_url }} yang dikirim oleh controller Laravel. Jika QRIS tersedia, pengguna dapat melihat kode, mengecek status pembayaran dengan tombol, dan mengetahui apakah pembayaran masih “pending” atau sudah berhasil. Jika tidak, muncul pesan bahwa QRIS belum tersedia. Desainnya modern, fokus pada keterbacaan dan kenyamanan pengguna.

e. Tombol simulasi dan copy URL



```
<a href="{{ $deeplink_url }}"
  id="deeplink-btn"
  class="inline-block bg-green-600 hover:bg-green-700 text-white px-8 py-3 rounded-lg font-medium transition shadow-lg hover:shadow-xl w-full sm:w-auto">
  Pay with Mobile App
<button id="copy-url-btn" class="mt-2 inline-block bg-indigo-600 hover:bg-indigo-700 text-white px-4 py-2 rounded-lg font-medium transition">
  Copy QRIS URL
<a href="https://simulator.sandbox.midtrans.com/v2/qris/index"
  target="_blank"
  rel="noopener noreferrer"
  class="inline-block bg-indigo-600 hover:bg-indigo-700 text-white px-8 py-3 rounded-lg font-medium transition shadow-lg hover:shadow-xl w-full sm:w-auto">
  Simulate Payment
```

Bagian ini menyediakan fitur tambahan untuk testing pembayaran, seperti membuka aplikasi gopay melalui deeplink, menyalin URL QRIS ke clipboard (copy-url-btn), dan mengakses simulator Midtrans Sandbox untuk uji coba tanpa transaksi nyata. Tombol-tombol ini penting pada tahap pengembangan sistem pembayaran untuk memastikan integrasi berjalan benar sebelum diterapkan secara produksi.

f. Script cek status pembayaran otomatis

```
<script>
  const orderId = "{{ $order_id }}";
  function checkPaymentStatus() {
    fetch(`/check-status/${orderId}`)
      .then(res => res.json())
      .then(data => {
        if (data.transaction_status === 'settlement') {
          statusText.textContent = 'Payment Success ✓';
        }
      });
  }
  setInterval(checkPaymentStatus, 10000);
</script>
```

Bagian javascript ini berfungsi mengecek status pembayaran secara otomatis ke server Laravel melalui endpoint /check-status/{order_id} setiap 10 detik. Jika status transaksi dari Midtrans sudah settlement, maka teks status akan berubah menjadi “Payment Success ✓”. Fitur polling ini membuat sistem dapat memantau pembayaran real-time tanpa perlu refresh manual oleh pengguna.

g. Sricpt copy URL ke clipboard

```
<script>
  document.getElementById('copy-url-btn').addEventListener('click', function() {
    navigator.clipboard.writeText("{{ $qris_url }}").then(() => {
      alert('QRIS URL berhasil disalin!');
    });
  });
</script>
```

Bagian ini menambahkan fungsi salin URL QRIS otomatis ke clipboard menggunakan API navigator.clipboard. Saat pengguna menekan tombol “Copy QRIS URL”, link akan tersalin ke clipboard dan menampilkan pesan keberhasilan. Fungsi ini mempermudah pengguna membagikan atau mengakses link QRIS dari perangkat lain tanpa harus mengetik manual.

5. Midtrans.php

```
27  return [
28      'server_key' => env(key: 'MIDTRANS_SERVER_KEY'),
29      'client_key' => env(key: 'MIDTRANS_CLIENT_KEY'),
30      'merchant_id' => env(key: 'MIDTRANS_MERCHANT_ID'),
31      'base_url' => env(key: 'MIDTRANS_BASE_URL'),
32      'is_production' => env(key: 'MIDTRANS_IS_PRODUCTION', default: false),
33  ];
```

Kode di atas merupakan file konfigurasi untuk integrasi Midtrans di Laravel, yang berfungsi menyimpan pengaturan penting agar sistem pembayaran dapat berjalan dengan aman dan fleksibel. Baris komentar pertama menjelaskan bahwa pengaturan seperti sandbox (testing) dan production (nyata) dapat dikendalikan melalui file .env. Kunci seperti MIDTRANS_SERVER_KEY, MIDTRANS_CLIENT_KEY, dan MIDTRANS_MERCHANT_ID diambil dari environment menggunakan fungsi env(), agar data sensitif tidak langsung ditulis di kode. Nilai base_url digunakan untuk menentukan alamat API Midtrans, sementara is_production berfungsi menentukan mode aplikasi — false untuk pengujian di sandbox, dan true untuk mode produksi. Pendekatan ini membuat sistem lebih aman, terstruktur, dan mudah dipindahkan antara lingkungan pengujian dan produksi, karena perubahan cukup dilakukan di file .env tanpa mengubah logika program.

6. .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:GW1TV8eszAJ74aXH1XXX378/ojkObnIWYFExfYGp8c4=
APP_DEBUG=true
APP_URL=  https://adducible-gaylene-dayfly.ngrok-free.dev
ASSET_URL=  https://adducible-gaylene-dayfly.ngrok-free.dev

APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=sqlite
SESSION_DRIVER=database
SESSION_LIFETIME=120

MAIL_MAILER=log
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

MIDTRANS_BASE_URL=https://api.sandbox.midtrans.com
MIDTRANS_SERVER_KEY=Mid-server-uWa0BMc0qXaij3TN-VZER6SG
MIDTRANS_CLIENT_KEY=Mid-client-LyxYd_yORlE-9EyV
MIDTRANS_MERCHANT_ID=GG314276913
MIDTRANS_IS_PRODUCTION=false
```

File .env ini merupakan berkas konfigurasi utama Laravel yang menyimpan semua pengaturan penting aplikasi, termasuk informasi server, database, sistem log, dan integrasi pihak ketiga seperti Midtrans. Bagian APP_ mengatur identitas aplikasi seperti nama (APP_NAME), lingkungan pengembangan (APP_ENV=local), kunci enkripsi (APP_KEY), serta URL utama yang menggunakan Ngrok agar bisa diakses publik untuk testing pembayaran. APP_DEBUG=true menampilkan pesan error detail saat pengembangan. Bagian DB_CONNECTION=sqlite menandakan bahwa aplikasi menggunakan database sqlite, yang sederhana dan cocok untuk pengujian lokal. Konfigurasi SESSION_DRIVER=database digunakan untuk menyimpan data sesi pengguna di database. Sementara itu, MAIL_MAILER=log berarti pengiriman email akan disimpan di log (tidak dikirim sungguhan).

Bagian paling penting adalah konfigurasi Midtrans, yaitu MIDTRANS_BASE_URL (alamat API sandbox), MIDTRANS_SERVER_KEY, MIDTRANS_CLIENT_KEY, dan MIDTRANS_MERCHANT_ID, yang merupakan kredensial agar aplikasi dapat terhubung dengan sistem pembayaran Midtrans. Nilai MIDTRANS_IS_PRODUCTION=false menunjukkan bahwa sistem masih berjalan dalam mode sandbox/testing, bukan transaksi nyata. Dengan adanya file .env ini, Laravel dapat memisahkan konfigurasi dari kode, membuat aplikasi lebih aman, fleksibel, dan mudah dipindahkan antar lingkungan (lokal, staging, hingga produksi).

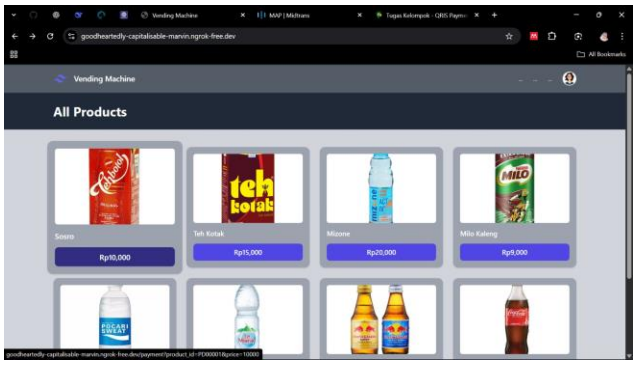
7. Web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PaymentController;

Route::get(uri: '/', action: [PaymentController::class, 'index']);

Route::get(uri: '/payment', action: [PaymentController::class, 'payment'])->name(name: 'payment');

Route::get(uri: '/check-status/{order_id}', action: [PaymentController::class, 'checkStatus']);
```

Kode di atas merupakan definisi routing dalam Laravel yang mengatur alur permintaan (request) pengguna menuju fungsi yang sesuai di dalam paymentcontroller. Pada baris pertama, use Illuminate\Support\Facades\Route; digunakan untuk memanggil facade Route milik Laravel yang berfungsi mendefinisikan rute aplikasi web. Selanjutnya, use App\Http\Controllers\paymentcontroller; berfungsi untuk memanggil controller yang akan menangani logika dari masing-masing rute. Rute pertama Route::get('/', [paymentcontroller::class, 'index']); menampilkan halaman utama aplikasi ketika pengguna membuka URL root (/). Rute kedua Route::get('/payment', [paymentcontroller::class, 'payment'])->name('payment'); mengarahkan ke fungsi payment() untuk menampilkan halaman pembayaran dan diberi nama "payment" agar mudah dipanggil dengan route('payment') di view atau controller. Rute ketiga Route::get('/check-status/{order_id}', [paymentcontroller::class, 'checkstatus']); digunakan untuk memeriksa status pembayaran berdasarkan order_id yang dikirim melalui URL, biasanya dipanggil secara otomatis melalui javascript untuk memantau transaksi pengguna secara real-time. Dengan demikian, kode ini berperan penting dalam menghubungkan frontend (tampilan pengguna) dengan backend logic (fungsi-fungsi pengolahan pembayaran).

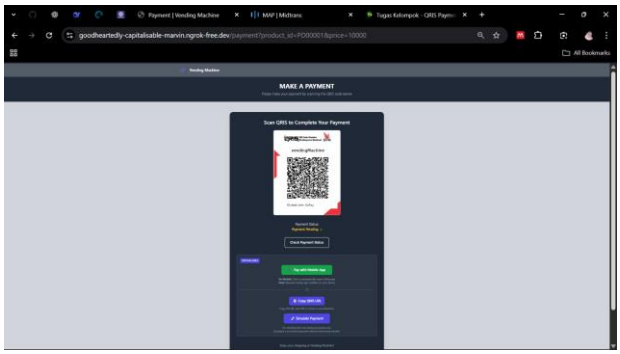
3.3. Hasil tampilan

1. /prepayment.blade.php

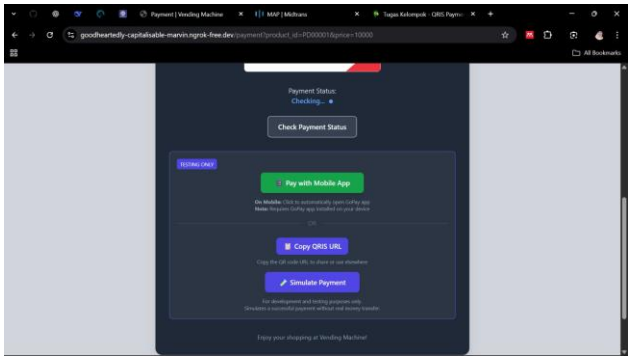


Menampilkan product + button harga direct ke (/payment.blade.php)

2. /payment.blade.php

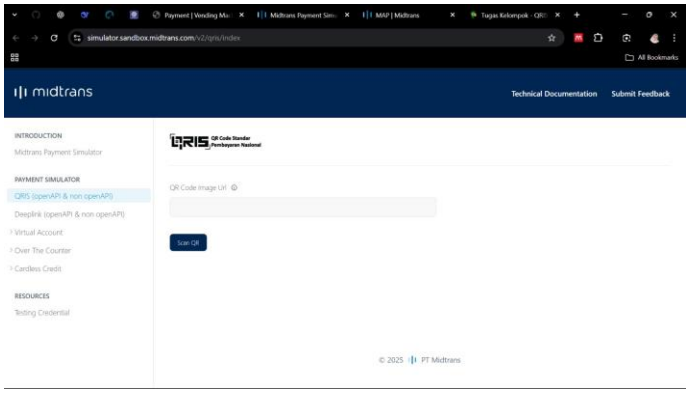


Menampilkan (core API midtrans) qris -> menampilkan status -> button 'cek payment status' -> button 'pay with mobile app' -> button copy (qris to url) -> button 'simulate payment' (ke web simulate payment milik midtrans)

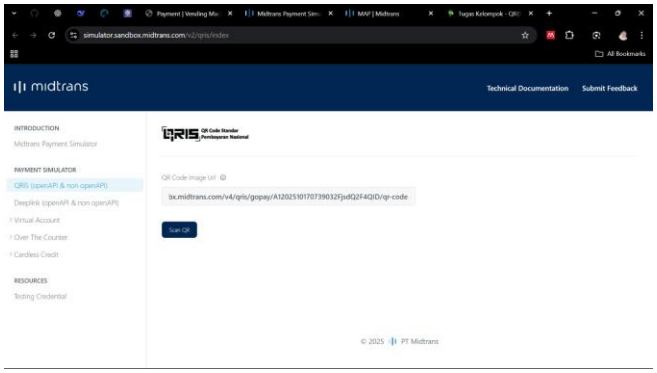


- kalo di klik button 'check payment status' -> status akan jadi 'checking..' dan imbuhan animasi berupa dot yang berdenyut
- kalo di klik button 'pay with mobile app' -> status akan berubah menjadi 'opening payment app'
- kalo udah 3 min setelah click button 'pay w mobile app' akan muncul status 'waiting for payment'
- copy url di button 'copy qris url' -> akan muncul 'copied' atau 'failed'

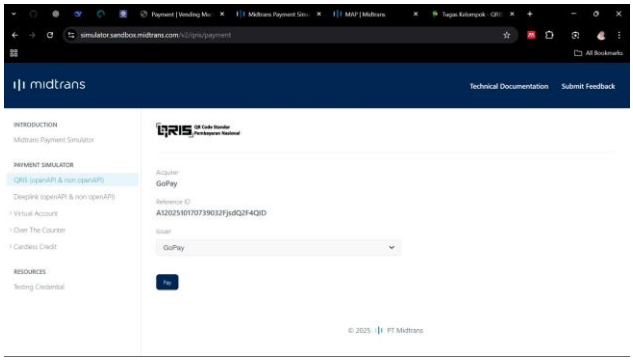
3. page simulate payment midtrans -> paste url tadi



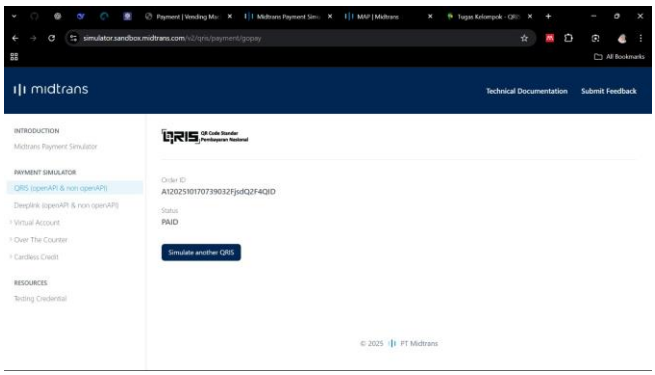
4. klik sqan qr



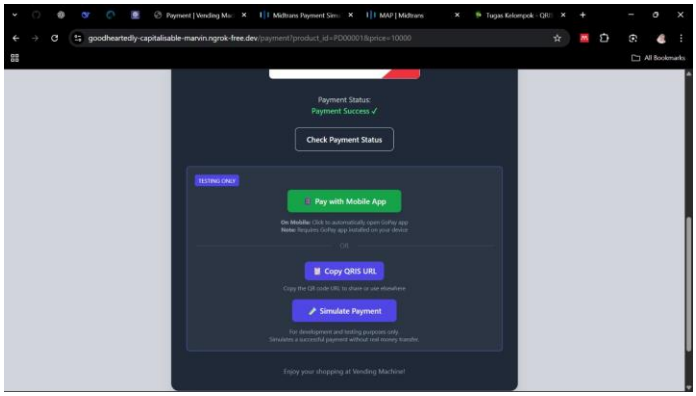
5. muncul informasi pembayaran -> klik 'pay'



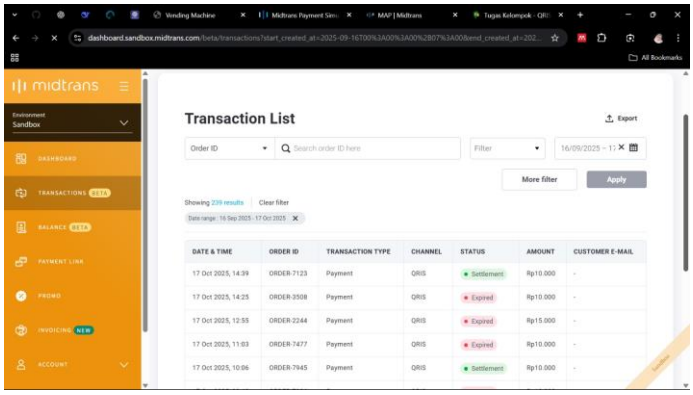
6. menampilkan status pembayaran



7. balik lagi ke /payment -> status jadi 'payment success'



8. tampilan transaksi di sandbox -> settlement artinya sukses



BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian, integrasi payment gateway menggunakan Laravel, Midtrans, dan Ngrok terbukti dapat berjalan dengan baik. Laravel berperan sebagai backend framework yang mengatur logika transaksi dan komunikasi API, Midtrans menyediakan layanan pembayaran yang aman dan cepat, sementara Ngrok membantu dalam proses pengujian webhook di lingkungan lokal. Dengan tambahan Tailwind CSS, tampilan antarmuka pembayaran menjadi lebih menarik dan mudah digunakan. Secara keseluruhan, kombinasi teknologi ini menghasilkan sistem pembayaran digital yang efisien, fleksibel, dan mudah diimplementasikan dalam pengembangan web modern.

4.2 Saran

Dalam pengembangan selanjutnya, disarankan untuk menggunakan mode production Midtrans agar sistem benar-benar siap diterapkan pada transaksi nyata. Penggunaan sertifikat SSL dan validasi data tambahan juga perlu diperhatikan untuk meningkatkan keamanan. Selain itu, sistem dapat dikembangkan lebih lanjut dengan menambahkan fitur riwayat transaksi, notifikasi email, serta integrasi dengan metode pembayaran lain seperti e-wallet dan virtual account.

DAFTAR PUSTAKA

Alamsyah, R., & Pratama, D. (2023). Analisis Integrasi QRIS Payment Gateway Menggunakan Laravel Framework. Jurnal Sistem dan Teknologi Informasi Indonesia, 12(1), 34–41.

Kumar, P., & Singh, R. (2021). API Security and Integration in Cloud-Based Systems. International Journal of Computer Science and Applications, 18(4), 45–53.

Laravel. (2024). Laravel Framework Documentation. Retrieved from https://laravel.com/docs

Midtrans. (2024). Midtrans API Documentation. Retrieved from https://docs.midtrans.com

Ngrok. (2023). Ngrok Documentation: Tunneling for Local Development. Retrieved from https://ngrok.com/docs

Sutrisno, A., & Rahman, H. (2022). Implementasi Payment Gateway Midtrans pada Aplikasi E-Commerce Berbasis Laravel. Jurnal Teknologi Informasi dan Sistem Informasi, 9(2), 89–97.

Tailwind Labs. (2024). Tailwind CSS Official Documentation. Retrieved from https://tailwindcss.com/docs