Sebentar, apa yang membedakan tabel payment dan transaction? Karena saya memang bingung akan 2 hal itu.

**ChatGPT said:**

Memang, tabel **payment** dan **transaction** dalam sistem pembayaran bisa terlihat mirip, tapi mereka sebenarnya **memiliki peran yang berbeda** dalam alur transaksi. Mari kita uraikan perbedaan mendasarnya:

**1. Tabel transaction**

Tabel **transaction** lebih fokus pada **proses transaksi secara keseluruhan**, yang berisi data terkait **transaksi yang dilakukan oleh pengguna**. Ini mencakup informasi yang **lebih luas** tentang transaksi.

**Tujuan:**

* **Menangani proses transaksi**: Mencatat informasi tentang transaksi seperti total harga, kode transaksi, tanggal transaksi, dan status transaksi.
* **Hubungan dengan pembayaran**: Setiap transaksi dapat memiliki satu atau lebih pembayaran yang terhubung.

**Fields yang umum ada di transaction:**

* **transaction\_id**: ID unik untuk setiap transaksi.
* **transaction\_code**: Kode unik transaksi, yang bisa digunakan untuk identifikasi atau pelacakan.
* **total\_price**: Jumlah total transaksi.
* **transaction\_status**: Status dari transaksi (misalnya, pending, paid, cancelled).
* **transaction\_date**: Tanggal dan waktu transaksi dibuat.
* **user\_id**: Menyimpan ID pengguna yang melakukan transaksi.
* **payment**: Relasi ke tabel payment, yang berhubungan dengan pembayaran untuk transaksi ini.

**Peran:**

Tabel **transaction** mencatat transaksi secara keseluruhan, yang bisa mencakup beberapa pembayaran. Misalnya, jika pengguna melakukan pembayaran cicilan atau memiliki beberapa metode pembayaran, **semua pembayaran** ini terhubung ke **satu transaksi**.

**2. Tabel payment**

Tabel **payment** adalah **detail pembayaran yang terkait dengan transaksi**. Setiap entri di tabel ini menggambarkan **satu kali pembayaran** yang dilakukan dalam satu transaksi.

**Tujuan:**

* **Mengelola pembayaran**: Mencatat informasi tentang masing-masing pembayaran dalam transaksi.
* **Hubungan dengan transaksi**: Setiap pembayaran terkait dengan satu transaksi, dan setiap transaksi bisa memiliki **beberapa pembayaran** (misalnya, jika pengguna membayar dalam beberapa tahap atau menggunakan beberapa metode).

**Fields yang umum ada di payment:**

* **payment\_id**: ID unik untuk setiap pembayaran.
* **transaction\_id**: Relasi ke ID transaksi yang terhubung dengan pembayaran ini.
* **payment\_status**: Status pembayaran (misalnya, pending, success, failed).
* **payment\_amount**: Jumlah uang yang dibayar pada pembayaran ini.
* **service\_fee**: Biaya layanan untuk pembayaran.
* **payment\_time**: Waktu pembayaran dilakukan.
* **payment\_method**: Metode pembayaran yang digunakan (misalnya, kartu kredit, transfer bank).
* **reference\_number**: Nomor referensi dari pembayaran.

**Peran:**

Tabel **payment** mencatat **setiap pembayaran individu** yang dilakukan untuk transaksi. Misalnya, transaksi pembelian tiket dapat terdiri dari beberapa pembayaran, dan masing-masing pembayaran ini akan tercatat di tabel **payment**.

**Contoh Alur yang Memperjelas Perbedaan:**

1. **Pengguna melakukan transaksi pembelian tiket**:
   * Tabel **transaction** mencatat keseluruhan transaksi pembelian tiket: ID transaksi, total harga, dan status transaksi (pending).
2. **Pengguna melakukan beberapa pembayaran**:
   * Mungkin pengguna memilih metode pembayaran **transfer bank** untuk setengah jumlah dan **e-wallet** untuk sisanya.
   * Setiap pembayaran akan dicatat di tabel **payment**, masing-masing dengan status pembayaran yang sesuai (misalnya, pending, success, atau failed).
3. **Hubungan antara transaction dan payment**:
   * Tabel **payment** akan mengacu pada **transaction\_id** untuk menunjukkan bahwa pembayaran tersebut terkait dengan transaksi tertentu.
   * **Satu transaksi** bisa memiliki banyak **pembayaran**, tetapi **setiap pembayaran hanya terkait dengan satu transaksi**.

**🧠 Kesimpulan:**

* **Tabel transaction** mencatat data **umum transaksi** (seperti harga total dan status transaksi).
* **Tabel payment** mencatat detail **pembayaran** yang dilakukan untuk transaksi tersebut (seperti jumlah yang dibayar, metode pembayaran, dan status).

Jadi, kamu membutuhkan kedua tabel ini karena mereka menangani dua **level informasi** yang berbeda. Tabel transaction mengelola transaksi itu sendiri, sementara tabel payment mengelola rincian pembayaran yang terkait dengan transaksi tersebut.

Apakah kamu ingin melanjutkan untuk menambahkan fitur **pembayaran Midtrans** di bagian ini atau ada hal lain yang masih belum jelas?

**You said:**

Sebelum membahas Midtrans. Bagaimana alur sistem secara keseluruhan dari schema.prisma yang saya berikan?

**ChatGPT said:**

Berdasarkan **schema.prisma** yang kamu berikan, saya akan jelaskan **alur sistem secara keseluruhan** untuk memberikan gambaran tentang bagaimana entitas-entitas tersebut berinteraksi dan mengalir dalam sistem.

**Alur Sistem Berdasarkan schema.prisma:**

1. **Pengguna dan Role (User Management)**
   * **Tabel users** menyimpan data pengguna, termasuk nama, email, password, dan nomor telepon.
   * **Relasi reviews** menunjukkan bahwa pengguna dapat memberikan ulasan pada spot wisata.
   * **Relasi transactions** menunjukkan bahwa pengguna dapat melakukan transaksi untuk pembelian tiket dan layanan.
2. **Kegiatan Wisata dan Fasilitas**
   * **Tabel spot\_tourism** menyimpan data tentang destinasi wisata seperti nama, deskripsi, alamat, dan rating. Setiap spot wisata dapat memiliki beberapa fasilitas yang dikelola melalui **facility\_tourism**.
   * **Tabel facility** menyimpan data fasilitas (seperti area parkir, toilet, kantin) yang dapat dimiliki oleh spot wisata. Fasilitas ini berhubungan dengan spot wisata melalui **facility\_tourism**.
3. **Katalog Tiket (Ticketing)**
   * **Tabel ticket** menyimpan data tentang tiket yang dijual untuk setiap spot wisata, termasuk harga tiket dan jumlah stok yang tersedia.
   * **Relasi transaction\_detail** menghubungkan antara **ticket** dan **transaction**, yang mencatat pembelian tiket oleh pengguna.
4. **Transaksi Pembelian Tiket**
   * **Tabel transaction** menyimpan data transaksi secara keseluruhan (misalnya, kode transaksi, total harga, status transaksi, dan tanggal transaksi). Transaksi ini bisa mencakup beberapa pembayaran dan tiket yang dibeli.
   * **Tabel transaction\_detail** menyimpan rincian tiket yang dibeli, termasuk jumlah tiket yang dibeli, harga per tiket, dan subtotal.
5. **Pembayaran**
   * **Tabel payment** berfungsi untuk mencatat pembayaran yang dilakukan untuk transaksi, termasuk jumlah yang dibayar, biaya layanan, status pembayaran, dan metode pembayaran yang digunakan.
   * **Tabel payment\_method** menyimpan berbagai metode pembayaran (seperti kartu kredit, transfer bank, dan e-wallet).
   * **Relasi payment\_provider** menyimpan informasi tentang penyedia layanan pembayaran (seperti Midtrans) yang mendukung berbagai metode pembayaran.
6. **Ulasan**
   * **Tabel reviews** menyimpan ulasan yang diberikan oleh pengguna pada spot wisata tertentu. Setiap ulasan berisi rating, deskripsi, dan waktu pembuatan.
7. **Operasional Spot Wisata**
   * **Tabel operating\_hours** menyimpan jam operasional setiap spot wisata berdasarkan hari. Data ini berguna untuk menampilkan waktu buka dan tutup spot wisata pada hari tertentu.
8. **Kategori Wisata**
   * **Tabel category\_tourism** menyimpan kategori spot wisata seperti Pantai, Gunung, Taman, dsb., yang digunakan untuk mengelompokkan spot wisata di aplikasi.
9. **Desa (Lokasi Wisata)**
   * **Tabel village** menyimpan data tentang desa yang menjadi lokasi spot wisata. Setiap desa dapat memiliki beberapa spot wisata yang dikelola.

**Alur Proses Sistem:**

Berdasarkan entitas-entitas di atas, berikut adalah alur proses keseluruhan yang menggambarkan bagaimana data mengalir dalam sistem:

1. **Pengguna Mendaftar dan Login**:
   * Pengguna mengisi data diri mereka (nama, email, dll) untuk mendaftar melalui **tabel users**.
   * Pengguna bisa melakukan login, yang menghubungkan aplikasi dengan session atau token autentikasi.
2. **Menelusuri Wisata**:
   * Pengguna dapat menjelajah **spot wisata** melalui fitur pencarian yang mengakses data dari **tabel spot\_tourism**.
   * Mereka dapat melihat fasilitas yang tersedia untuk setiap spot wisata, yang datanya ada di **tabel facility**.
3. **Memilih Tiket**:
   * Pengguna memilih tiket untuk spot wisata yang mereka pilih. Data tiket ini diambil dari **tabel ticket**, dan tiket akan terhubung ke **tabel transaction\_detail** setelah pembelian.
4. **Melakukan Transaksi Pembelian**:
   * Pengguna membuat transaksi pembelian tiket melalui **tabel transaction**. Transaksi ini mencatat total harga dan status transaksi.
   * Setiap transaksi dapat mencakup satu atau lebih tiket, yang dirinci di **tabel transaction\_detail**.
5. **Melakukan Pembayaran**:
   * Setelah memilih tiket, pengguna akan dibawa untuk melakukan pembayaran. **Tabel payment** digunakan untuk mencatat pembayaran yang dilakukan oleh pengguna untuk transaksi.
   * Data pembayaran mencakup **payment\_status**, **payment\_amount**, dan **payment\_method**.
6. **Notifikasi Pembayaran**:
   * Pembayaran diproses menggunakan penyedia layanan seperti **Midtrans**, yang dapat digunakan untuk memverifikasi dan mengonfirmasi status pembayaran.
   * Setelah pembayaran berhasil, status transaksi di **tabel transaction** diperbarui menjadi paid.
7. **Ulasan dan Rating**:
   * Setelah transaksi selesai, pengguna dapat memberikan **ulasan** pada spot wisata yang mereka kunjungi, yang akan disimpan di **tabel reviews**.
   * Ulasan ini mencakup rating dan deskripsi, yang akan ditampilkan pada halaman detail spot wisata.

**Ringkasan Alur Sistem:**

1. **User** → **users** (registrasi dan login)
2. **Spot Wisata** → **spot\_tourism** (akses destinasi wisata)
3. **Fasilitas** → **facility** (fasilitas yang tersedia di spot)
4. **Tiket** → **ticket** (pemilihan tiket)
5. **Transaksi** → **transaction** (mencatat pembelian tiket)
6. **Pembayaran** → **payment** (melakukan pembayaran untuk transaksi)
7. **Ulasan** → **reviews** (memberikan feedback dan rating spot wisata)

**Kebutuhan Selanjutnya:**

* **API Pembayaran (Midtrans)**: Pengguna dapat melakukan pembayaran melalui Midtrans, yang harus mencakup transaksi, metode pembayaran, dan callback status pembayaran.
* **Pengelolaan Status Pembayaran**: Setelah pembayaran dilakukan, sistem harus dapat memverifikasi dan mengonfirmasi status pembayaran.

Jika kamu ingin melanjutkan ke integrasi **Midtrans** atau mengerjakan bagian lain dari sistem ini, beri tahu saya, dan saya akan bantu melanjutkan!

**You said:**

Karena saya pasti menggunakan Midtrans, apakah perlu adanya tabel payment, payment\_method, dan payment\_provider? Karena saya masih kebingungan dengan 3 tabel itu.

**ChatGPT said:**

Memahami kebingungan kamu mengenai tabel **payment**, **payment\_method**, dan **payment\_provider**, mari kita ulas satu per satu untuk memastikan apakah tabel-tabel ini diperlukan dalam integrasi dengan **Midtrans** atau apakah bisa disederhanakan.

**1. Tabel payment (Peran Utama: Mencatat Pembayaran)**

Tabel **payment** bertanggung jawab untuk mencatat semua detail pembayaran yang dilakukan oleh pengguna. Ini adalah tempat utama untuk menyimpan informasi yang berhubungan langsung dengan transaksi pembayaran.

**Apa yang disimpan di payment?**

* **payment\_id**: ID unik untuk setiap entri pembayaran.
* **transaction\_id**: Relasi ke transaksi yang sedang dibayar (misalnya, satu transaksi bisa memiliki satu atau lebih pembayaran jika ada cicilan).
* **payment\_status**: Status pembayaran seperti pending, success, atau failed.
* **payment\_amount**: Jumlah uang yang dibayar pada pembayaran ini.
* **service\_fee**: Biaya layanan yang dikenakan pada pembayaran.
* **total\_payment**: Jumlah total yang dibayar, termasuk biaya layanan.
* **payment\_time**: Waktu pembayaran dilakukan.
* **payment\_method**: Metode pembayaran (misalnya, kartu kredit, bank transfer, dll).
* **response\_data**: Menyimpan data respons dari Midtrans (atau penyedia pembayaran lainnya).

**Peran payment dalam sistem:**

Tabel **payment** penting karena ia mencatat **proses pembayaran yang terkait dengan transaksi**. Misalnya, setelah pengguna memilih produk/tiket dan melakukan pembayaran, detail pembayaran tersebut akan disimpan di sini.

**✅ Apakah perlu?**

Ya, tabel **payment** diperlukan untuk menyimpan **informasi pembayaran** yang telah dilakukan oleh pengguna. Ini sangat berguna ketika kamu perlu melacak status pembayaran atau menampilkan riwayat pembayaran pengguna.

**2. Tabel payment\_method (Metode Pembayaran yang Tersedia)**

Tabel **payment\_method** mencatat **metode pembayaran** yang dapat digunakan oleh pengguna untuk melakukan pembayaran. Ini mencakup informasi tentang berbagai jenis metode pembayaran yang didukung, seperti **kartu kredit**, **bank transfer**, **e-wallet**, atau **QRIS**.

**Apa yang disimpan di payment\_method?**

* **method\_id**: ID unik untuk metode pembayaran.
* **method\_name**: Nama metode pembayaran (misalnya, Credit Card, Bank Transfer, QRIS).
* **method\_code**: Kode unik yang digunakan untuk mengidentifikasi metode pembayaran.
* **method\_icon**: Ikon yang terkait dengan metode pembayaran (untuk UI).
* **service\_fee**: Biaya layanan tambahan yang terkait dengan metode pembayaran.
* **minimum\_payment**: Pembayaran minimum yang diperlukan untuk menggunakan metode ini.
* **maximum\_payment**: Pembayaran maksimum yang dapat dilakukan dengan metode ini.
* **is\_active**: Menandakan apakah metode pembayaran aktif atau tidak.

**Peran payment\_method dalam sistem:**

Tabel **payment\_method** memberikan fleksibilitas dalam mengelola berbagai **metode pembayaran** yang dapat dipilih pengguna. Kamu bisa menambahkan berbagai jenis metode pembayaran dari Midtrans, seperti kartu kredit, bank transfer, atau e-wallet, dan setiap transaksi akan memiliki metode pembayaran yang terkait.

**✅ Apakah perlu?**

Jika kamu hanya ingin mendukung pembayaran dengan **Midtrans**, kamu tidak perlu membuat tabel payment\_method yang rumit. **Midtrans** sudah menyediakan API yang dapat menangani berbagai metode pembayaran (kartu kredit, e-wallet, transfer bank, dll). Namun, tabel ini masih bisa berguna jika kamu ingin:

* **Mendukung berbagai metode pembayaran** selain yang disediakan oleh Midtrans.
* **Menampilkan opsi pembayaran kepada pengguna** di UI dan melacaknya di database.

Jika kamu hanya akan menggunakan Midtrans dengan sedikit variasi metode pembayaran, tabel ini mungkin tidak perlu terlalu banyak data dan bisa cukup sederhana.

**3. Tabel payment\_provider (Penyedia Pembayaran)**

Tabel **payment\_provider** mencatat **informasi tentang penyedia pembayaran** (misalnya, **Midtrans**, **Doku**, **Xendit**, dll). Ini berguna jika kamu berencana untuk mendukung **lebih dari satu penyedia pembayaran**.

**Apa yang disimpan di payment\_provider?**

* **provider\_id**: ID unik untuk penyedia pembayaran.
* **provider\_name**: Nama penyedia pembayaran (misalnya, Midtrans).
* **provider\_logo**: Logo penyedia pembayaran.
* **provider\_desc**: Deskripsi penyedia pembayaran.
* **is\_active**: Menandakan apakah penyedia pembayaran aktif atau tidak.

**Peran payment\_provider dalam sistem:**

Tabel **payment\_provider** berguna jika kamu berencana untuk mengintegrasikan **beberapa penyedia pembayaran** dalam satu aplikasi. Namun, jika hanya menggunakan **Midtrans**, tabel ini bisa jadi **tidak diperlukan**, karena kamu hanya akan memiliki satu penyedia pembayaran yang digunakan.

**✅ Apakah perlu?**

Jika kamu **hanya menggunakan Midtrans** dan tidak berniat menambah penyedia pembayaran lain, maka **payment\_provider** bisa dihilangkan atau disederhanakan, karena hanya akan mengarah ke satu penyedia (Midtrans).

**🎯 Ringkasan Keperluan Tabel-tabel ini:**

| **Tabel** | **Keterangan** | **Perlu atau Tidak?** |
| --- | --- | --- |
| **payment** | Menyimpan data pembayaran (satu transaksi bisa memiliki satu atau lebih pembayaran). | **Perlu** |
| **payment\_method** | Menyimpan berbagai metode pembayaran (misalnya, kartu kredit, bank transfer, e-wallet). | **Perlu** jika ada berbagai metode yang didukung (misalnya selain Midtrans). Jika hanya Midtrans, bisa disederhanakan. |
| **payment\_provider** | Menyimpan penyedia pembayaran (misalnya, Midtrans, Doku, Xendit). | **Tidak perlu** jika hanya menggunakan Midtrans. |

**💡 Simpulan:**

* **payment** adalah **wajib** karena mencatat pembayaran yang terjadi untuk transaksi.
* **payment\_method** dan **payment\_provider** adalah **opsional**. Jika kamu hanya menggunakan **Midtrans** untuk pembayaran, kamu bisa mempertimbangkan untuk **menyederhanakan** atau **menghapus** tabel ini.

Apakah penjelasan ini sudah cukup jelas, dan apakah kamu ingin melanjutkan dengan implementasi **Midtrans** atau ada pertanyaan lainnya?

Jadi bagaimana dengan jika saya memang menggunakan Midtrans, apakah perlu ada revisi pada 3 tabel tersebut, atau ada revisi total untuk semua tabel di database saya? schema.prisma (database baturraden\_tourism)

|  |
| --- |
| generator client {    provider = "prisma-client-js"  }  datasource db {    provider = "mysql"    url      = env("DATABASE\_URL")  }  model category\_tourism {    category\_id   String         @id @db.Char(5)    category\_name String         @db.VarChar(20)    category\_desc String?        @db.VarChar(150)    category\_icon String?        @db.VarChar(255)    spots         spot\_tourism[]  }  model facility {    facility\_id       String             @id @db.Char(10)    facility\_name     String             @db.VarChar(100)    facility\_icon     String?            @db.VarChar(255)    facility\_tourisms facility\_tourism[]  }  model facility\_tourism {    facility\_tourism\_id String        @id @db.Char(10)    spot\_id             String?       @db.Char(10)    facility\_id         String?       @db.Char(10)    spot                spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "facility\_tourism\_ibfk\_1")    facility            facility?     @relation(fields: [facility\_id], references: [facility\_id], onDelete: Cascade, onUpdate: NoAction, map: "facility\_tourism\_ibfk\_2")    @@index([facility\_id], map: "facility\_id")    @@index([spot\_id], map: "spot\_id")  }  model gallery {    gallery\_id      String        @id @db.Char(10)    spot\_id         String?       @db.Char(10)    gallery\_img     String?       @db.VarChar(255)    gallery\_caption String?       @db.VarChar(255)    spot            spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "gallery\_ibfk\_1")    @@index([spot\_id], map: "spot\_id")  }  model operating\_hours {    operating\_id  String        @id @db.Char(10)    spot\_id       String?       @db.Char(10)    operating\_day OperatingDay    hours\_open    DateTime      @db.Time(0)    hours\_closed  DateTime      @db.Time(0)    spot          spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "operating\_hours\_ibfk\_1")    @@index([spot\_id], map: "spot\_id")  }  model payment {    payment\_id       Int             @id @default(autoincrement())    transaction\_id   Int?    method\_id        Int?    reference\_number String          @unique(map: "reference\_number") @db.VarChar(50)    payment\_amount   Decimal         @db.Decimal(10, 2)    service\_fee      Decimal         @db.Decimal(10, 2)    total\_payment    Decimal?        @db.Decimal(10, 2)    payment\_status   PaymentStatus?  @default(pending)    payment\_time     DateTime?       @default(now()) @db.Timestamp(0)    expiration\_time  DateTime?       @db.Timestamp(0)    response\_data    Json?    transaction      transaction?    @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction, map: "payment\_ibfk\_1")    method           payment\_method? @relation(fields: [method\_id], references: [method\_id], onUpdate: NoAction, map: "payment\_ibfk\_2")    @@index([method\_id], map: "method\_id")    @@index([transaction\_id], map: "transaction\_id")  }  model payment\_method {    method\_id       Int               @id @default(autoincrement())    provider\_id     String?           @db.Char(10)    method\_name     String            @db.VarChar(50)    method\_code     String            @unique(map: "method\_code") @db.VarChar(50)    method\_icon     String?           @db.VarChar(255)    service\_fee     Decimal           @default(0.00) @db.Decimal(10, 2)    minimum\_payment Decimal           @default(0.00) @db.Decimal(10, 2)    maximum\_payment Decimal           @db.Decimal(10, 2)    is\_active       Boolean?          @default(true)    payments        payment[]    provider        payment\_provider? @relation(fields: [provider\_id], references: [provider\_id], onUpdate: NoAction, map: "payment\_method\_ibfk\_1")    @@index([provider\_id], map: "provider\_id")  }  model payment\_provider {    provider\_id   String           @id @db.Char(10)    provider\_name String           @db.VarChar(50)    provider\_desc String?          @db.VarChar(255)    provider\_logo String?          @db.VarChar(255)    is\_active     Boolean?         @default(true)    methods       payment\_method[]  }  model reviews {    reviews\_id         String        @id @db.Char(10)    spot\_id            String?       @db.Char(10)    user\_id            String?       @db.Char(10)    reviews\_rating     Int?          @db.TinyInt    reviews\_desc       String?       @db.Text    reviews\_created\_at DateTime?     @default(now()) @db.Timestamp(0)    spot               spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "reviews\_ibfk\_1")    user               users?        @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade, onUpdate: NoAction, map: "reviews\_ibfk\_2")    @@index([spot\_id], map: "spot\_id")    @@index([user\_id], map: "user\_id")  }  model spot\_tourism {    spot\_id           String             @id @db.Char(10)    category\_id       String?            @db.Char(5)    village\_id        String?            @db.Char(10)    spot\_name         String             @db.VarChar(60)    spot\_desc         String?            @db.Text    spot\_address      String?            @db.VarChar(200)    spot\_maps         String?            @db.VarChar(255)    spot\_thumbnail    String?            @db.VarChar(100)    spot\_rating       Decimal?           @default(0.00) @db.Decimal(3, 2)    spot\_contact      String?            @db.VarChar(15)    facility\_tourisms facility\_tourism[]    galleries         gallery[]    operating\_hours   operating\_hours[]    reviews           reviews[]    category          category\_tourism?  @relation(fields: [category\_id], references: [category\_id], onUpdate: NoAction, map: "spot\_tourism\_ibfk\_1")    village           village?           @relation(fields: [village\_id], references: [village\_id], onUpdate: NoAction, map: "spot\_tourism\_ibfk\_2")    tickets           ticket[]    @@index([category\_id], map: "category\_id")    @@index([village\_id], map: "village\_id")  }  model ticket {    ticket\_id           String               @id @db.Char(20)    spot\_id             String?              @db.Char(10)    ticket\_name         String?              @db.VarChar(100)    ticket\_price        Decimal              @db.Decimal(10, 2)    ticket\_desc         String?              @db.Text    ticket\_stock        Int?                 @default(0)    code                String               @db.VarChar(10)    url\_qr              String               @db.Text    spot                spot\_tourism?        @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "ticket\_ibfk\_1")    transaction\_details transaction\_detail[]    @@index([spot\_id], map: "spot\_id")  }  model transaction {    transaction\_id      Int                  @id @default(autoincrement())    user\_id             String?              @db.Char(10)    transaction\_code    String               @unique(map: "transaction\_code") @db.VarChar(50)    total\_price         Decimal              @db.Decimal(10, 2)    transaction\_status  TransactionStatus?   @default(pending)    transaction\_date    DateTime?            @default(now()) @db.Timestamp(0)    visit\_date          DateTime             @db.Date    payments            payment[]    user                users?               @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_ibfk\_1")    transaction\_details transaction\_detail[]    @@index([user\_id], map: "user\_id")  }  model transaction\_detail {    transaction\_detail\_id Int          @id @default(autoincrement())    transaction\_id        Int?    ticket\_id             String?      @db.Char(20)    detail\_quantity       Int    detail\_price          Decimal      @db.Decimal(10, 2)    detail\_subtotal       Decimal?     @db.Decimal(10, 2)    is\_used                TicketStatus @default(unverified) // Gunakan enum di sini    transaction           transaction? @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_detail\_ibfk\_1")    ticket                ticket?      @relation(fields: [ticket\_id], references: [ticket\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_detail\_ibfk\_2")    @@index([ticket\_id], map: "ticket\_id")    @@index([transaction\_id], map: "transaction\_id")  }  model users {    user\_id      String        @id @db.Char(10)    name         String        @db.VarChar(100)    email        String        @unique(map: "email") @db.VarChar(100)    password     String        @db.VarChar(255)    phone\_number String?       @db.VarChar(20)    address      String?       @db.Text    role         Role    reviews      reviews[]    transactions transaction[]  }  model village {    village\_id   String         @id @db.Char(10)    village\_name String         @db.VarChar(100)    village\_city String         @db.VarChar(100)    spots        spot\_tourism[]  }  enum OperatingDay {    Senin    Selasa    Rabu    Kamis    Jumat    Sabtu    Minggu  }  enum PaymentStatus {    pending    success    failed  }  enum TransactionStatus {    pending    paid    cancelled  }  enum Role {    admin    user    tourism\_owner  }  enum TicketStatus {    unverified    verified  } |

Dari **schema.prisma** yang kamu berikan, kita bisa lihat bahwa tabel **payment**, **payment\_method**, dan **payment\_provider** memiliki **peran khusus** dalam menangani transaksi pembayaran. Sekarang mari kita evaluasi apakah **perlu ada revisi** atau tidak pada ketiga tabel tersebut jika kamu hanya menggunakan **Midtrans**.

**Peran Tabel-Tabel dalam Proses Pembayaran:**

**1. Tabel payment**

Tabel **payment** adalah tabel yang mencatat **detail pembayaran** terkait dengan transaksi. Setiap entri mencatat informasi terkait pembayaran yang dilakukan untuk transaksi tertentu.

**Kebutuhan untuk payment:**

* **payment\_id**: ID unik untuk setiap pembayaran.
* **transaction\_id**: Relasi ke transaksi yang sedang dibayar (satu transaksi bisa memiliki beberapa pembayaran).
* **payment\_status**: Status pembayaran seperti pending, success, atau failed.
* **payment\_time**: Waktu pembayaran dilakukan.
* **payment\_method**: Metode pembayaran yang digunakan, seperti kartu kredit atau e-wallet.
* **response\_data**: Data respons dari Midtrans (untuk menyimpan informasi lebih lanjut tentang transaksi pembayaran, seperti URL untuk pembayaran, kode transaksi, dll.).

**Apakah perlu?**

* Ya, **payment tetap diperlukan** karena mencatat **detail pembayaran** yang dilakukan untuk setiap transaksi. Jika kamu menggunakan Midtrans untuk memproses pembayaran, kamu tetap perlu mencatat status pembayaran dan menghubungkannya dengan transaksi tertentu.

**2. Tabel payment\_method**

Tabel **payment\_method** menyimpan informasi tentang **metode pembayaran** yang dapat dipilih pengguna, seperti **kartu kredit**, **transfer bank**, **QRIS**, dan lain-lain.

**Kebutuhan untuk payment\_method:**

* **method\_id**: ID unik untuk metode pembayaran.
* **method\_name**: Nama metode pembayaran, seperti Credit Card, Bank Transfer, QRIS.
* **method\_code**: Kode unik untuk metode pembayaran.
* **is\_active**: Status apakah metode pembayaran aktif atau tidak.

**Apakah perlu?**

* Jika kamu hanya menggunakan **Midtrans** dan **metode pembayaran yang disediakan oleh Midtrans**, tabel ini **tidak terlalu diperlukan** dalam bentuk kompleks. Kamu hanya perlu mencatat **metode pembayaran yang didukung** oleh Midtrans, dan bisa langsung menghubungkannya ke transaksi.
* Namun, jika kamu ingin **menambah metode pembayaran lain** selain yang disediakan oleh Midtrans (misalnya, beberapa opsi tambahan di masa depan), kamu bisa menggunakan tabel ini untuk mempermudah pengelolaan dan menampilkan opsi pembayaran kepada pengguna.

**3. Tabel payment\_provider**

Tabel **payment\_provider** menyimpan informasi tentang penyedia **metode pembayaran**, seperti **Midtrans**.

**Kebutuhan untuk payment\_provider:**

* **provider\_id**: ID unik untuk penyedia layanan pembayaran.
* **provider\_name**: Nama penyedia layanan pembayaran (misalnya, Midtrans).
* **is\_active**: Status apakah penyedia pembayaran aktif atau tidak.

**Apakah perlu?**

* Jika kamu hanya menggunakan **Midtrans**, tabel **payment\_provider** bisa jadi **tidak diperlukan**. Semua data terkait penyedia (Midtrans) dapat langsung disimpan di dalam tabel **payment\_method**.
* **Jika ingin mendukung lebih dari satu penyedia pembayaran**, misalnya **Midtrans**, **Doku**, atau **Xendit**, tabel ini akan berguna untuk mencatat penyedia layanan pembayaran yang digunakan.

**Rekomendasi Revisi Tabel-Tabel:**

1. **Tabel payment**: **Tetap diperlukan**. Ini sangat penting untuk mencatat semua detail pembayaran, termasuk data dari Midtrans (status, waktu, dan amount pembayaran).
2. **Tabel payment\_method**: Jika hanya **Midtrans** yang digunakan, kamu bisa **menyederhanakan** tabel ini atau bahkan **menghapusnya**, dan langsung simpan metode pembayaran (misalnya, credit\_card, bank\_transfer, dll.) di **tabel payment**. Namun, jika berencana menambah berbagai metode pembayaran lainnya, tabel ini akan tetap berguna.
3. **Tabel payment\_provider**: Jika hanya menggunakan **Midtrans**, tabel ini **tidak diperlukan** dan bisa **dihapus**. Namun, jika ada rencana untuk menambah penyedia pembayaran lain, kamu bisa **mempertahankan** tabel ini.

**Revisi Total pada Tabel-tabel (Jika Hanya Menggunakan Midtrans):**

Jika kamu hanya berencana menggunakan **Midtrans**, kamu bisa melakukan beberapa **penyederhanaan** berikut:

**1. Simpan hanya metode pembayaran yang digunakan:**

* Di **tabel payment\_method**, kamu bisa hanya mencatat beberapa opsi pembayaran yang disediakan oleh Midtrans (misalnya, credit\_card, bank\_transfer, dan e\_wallet).

**2. Menghapus payment\_provider jika hanya menggunakan satu penyedia (Midtrans).**

**Contoh Penyederhanaan Struktur Tabel payment dan payment\_method:**

1. **Tabel payment\_method** (jika hanya satu penyedia seperti Midtrans):

|  |
| --- |
| prisma  model payment\_method {  method\_id Int @id @default(autoincrement())  method\_name String @db.VarChar(50)  method\_code String @db.VarChar(50)  is\_active Boolean? @default(true)  payments payment[] // Relasi ke tabel payment  } |

1. **Tabel payment** (sederhana, hanya mencatat informasi yang dibutuhkan):

|  |
| --- |
| prisma  model payment {  payment\_id Int @id @default(autoincrement())  transaction\_id Int?  method\_id Int?  reference\_number String @unique(map: "reference\_number") @db.VarChar(50)  payment\_amount Decimal @db.Decimal(10, 2)  service\_fee Decimal @db.Decimal(10, 2)  total\_payment Decimal? @db.Decimal(10, 2)  payment\_status PaymentStatus? @default(pending)  payment\_time DateTime? @default(now()) @db.Timestamp(0)  payment\_method payment\_method? @relation(fields: [method\_id], references: [method\_id])  transaction transaction? @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction)  } |

**Kesimpulan:**

* **Tabel payment** tetap diperlukan untuk mencatat detail pembayaran.
* **Tabel payment\_method** dapat disederhanakan atau dipertahankan jika kamu berencana untuk menambah berbagai metode pembayaran selain Midtrans.
* **Tabel payment\_provider** bisa **dihapus** jika hanya menggunakan **Midtrans**.

Apakah kamu ingin saya bantu dengan implementasi **Midtrans API** atau melanjutkan ke bagian lain?

Saya memang memiliki rasa penasaran cukup tinggi, jadi mengapa payment\_method ini disederhakan dan payment\_provider ini dihilangkan jika saya memang menggunakan Midtrans?

**ChatGPT said:**

Itu adalah pertanyaan yang sangat bagus! Memang, untuk sistem pembayaran seperti **Midtrans**, ada alasan mengapa saya menyarankan untuk **menyederhanakan** atau **menghilangkan** beberapa tabel seperti payment\_method dan payment\_provider. Mari kita jelaskan alasan di baliknya:

**1. Tabel payment\_method**

Tabel **payment\_method** biasanya digunakan untuk menyimpan berbagai metode pembayaran yang tersedia untuk pengguna, seperti:

* **Kartu Kredit**
* **Bank Transfer**
* **E-wallet**
* **QRIS**, dll.

Namun, jika kamu **hanya menggunakan Midtrans** sebagai penyedia pembayaran utama, maka semua metode pembayaran yang ingin kamu tawarkan kepada pengguna **sudah disediakan oleh Midtrans** melalui API mereka.

**Kenapa bisa disederhanakan?**

1. **Midtrans sudah mengelola berbagai metode pembayaran untukmu**. Misalnya, jika pengguna memilih **kartu kredit**, **bank transfer**, atau **e-wallet**, Midtrans akan menangani semua ini melalui satu API yang sama. Oleh karena itu, **tidak perlu tabel payment\_method yang menyimpan data metode pembayaran** secara manual.
2. **Informasi tentang metode pembayaran** (seperti kartu kredit, bank transfer, dll.) dapat langsung diambil dari **Midtrans API** saat transaksi dilakukan. Kamu bisa mendapatkan informasi tentang metode pembayaran yang digunakan langsung dari Midtrans (misalnya, menggunakan payment\_type di respons transaksi).

**💡 Solusi Penyederhanaan:**

Kamu bisa langsung menyimpan metode pembayaran **di dalam tabel payment** atau hanya menyimpan **payment\_status** dan informasi terkait, tanpa memerlukan tabel terpisah untuk metode pembayaran.

**2. Tabel payment\_provider**

Tabel **payment\_provider** menyimpan informasi tentang penyedia layanan pembayaran (misalnya **Midtrans**, **Doku**, **Xendit**, dll.). Biasanya, tabel ini berguna ketika kamu ingin mendukung lebih dari satu penyedia pembayaran.

**Kenapa bisa dihilangkan?**

Jika kamu hanya berencana **menggunakan satu penyedia pembayaran**, yaitu **Midtrans**, maka tabel **payment\_provider** **tidak diperlukan**. Midtrans sudah menyederhanakan seluruh proses pembayaran, jadi kamu tidak memerlukan tabel yang memisahkan informasi tentang penyedia layanan pembayaran.

1. **Satu penyedia pembayaran**: Jika hanya ada **Midtrans** sebagai penyedia pembayaran, tidak perlu memisahkan data penyedia pembayaran di tabel berbeda.
2. **Keuntungan dari menghapusnya**: Mengurangi kerumitan dan kompleksitas database jika hanya menggunakan satu penyedia. Cukup simpan informasi metode pembayaran di dalam tabel **payment\_method** dan **payment**, yang mengarah langsung ke transaksi.

**Apakah Harus Menggunakan Pendekatan ini?**

Pendekatan ini akan **lebih efisien** dalam hal penyimpanan dan pengelolaan data, karena kamu hanya akan menyimpan informasi yang **relevan** dengan Midtrans. Jika di masa depan kamu **berencana untuk menambahkan lebih banyak penyedia pembayaran**, kamu bisa **menambahkannya** kembali dengan mudah.

Namun, jika kamu **membutuhkan fleksibilitas** untuk mendukung beberapa penyedia pembayaran sejak awal, kamu bisa **mempertahankan tabel payment\_method dan payment\_provider**, dan nanti bisa menambahkan logika untuk menangani lebih dari satu penyedia.

**Contoh Alur Penggunaan Midtrans tanpa payment\_method dan payment\_provider:**

1. **Pembayaran dimulai** → **Midtrans API** menerima pembayaran dan memprosesnya.
2. **Status pembayaran** disimpan di **payment** dengan informasi seperti payment\_status, payment\_amount, dll.
3. Jika diperlukan, kamu bisa **mendapatkan informasi pembayaran** melalui API Midtrans, tanpa perlu menyimpan metode pembayaran secara terpisah.

**Ringkasan:**

* Jika **hanya menggunakan Midtrans**, kamu bisa **menyederhanakan** dan menghindari kompleksitas dengan menggabungkan data **metode pembayaran** langsung di dalam tabel **payment** dan **payment\_method**.
* **payment\_provider** bisa dihilangkan karena hanya ada satu penyedia pembayaran (Midtrans).
* **Mengurangi redundansi data** dan membuat struktur lebih mudah dikelola jika kamu hanya berencana menggunakan **Midtrans**.

Jika kamu ingin melanjutkan ke **integrasi Midtrans** atau **perubahan lain pada struktur database** yang lebih efisien, beri tahu saya!

Jika saya menggunakan Midtrans, apakah perlu menggunakan Ngrok?

**ChatGPT said:**

Menggunakan **Ngrok** tergantung pada kebutuhan kamu dalam **proses pengembangan** dan **integrasi dengan Midtrans**. Ngrok adalah alat yang digunakan untuk membuat **tunnel** ke localhost kamu, memungkinkan **akses publik** ke aplikasi yang sedang berjalan di lingkungan pengembangan lokal.

**Kapan Ngrok Dibutuhkan?**

1. **Integrasi dengan API eksternal seperti Midtrans**:
   * Jika kamu mengembangkan aplikasi lokal dan perlu menghubungkan aplikasi tersebut dengan API eksternal seperti **Midtrans** (yang memerlukan URL publik untuk callback), **Ngrok** bisa sangat berguna.
   * **Midtrans** membutuhkan **URL yang dapat diakses dari luar** untuk melakukan **callback** setelah pembayaran berhasil dilakukan. Jika kamu hanya mengembangkan aplikasi di localhost, **Midtrans tidak bisa mengakses localhost** secara langsung.
   * **Ngrok** memberikan **URL publik sementara** untuk aplikasi lokal, sehingga Midtrans bisa mengirimkan callback ke URL yang dapat dijangkau.

**Bagaimana Ngrok Bekerja dengan Midtrans?**

* **Callback URL** dari Midtrans memerlukan URL yang dapat diakses oleh **Midtrans** untuk memberitahukan status pembayaran (misalnya, ketika pembayaran berhasil atau gagal).
* **Ngrok** membuat URL publik sementara yang merutekan request ke server lokal kamu.
* Setelah kamu menjalankan Ngrok, kamu akan mendapatkan URL yang bisa digunakan di **Midtrans Dashboard** untuk mengonfigurasi **Callback URL**.

**Langkah-Langkah Menggunakan Ngrok dengan Midtrans:**

1. **Install Ngrok**:
   * Kamu bisa mengunduh dan menginstall Ngrok dari ngrok.com.
   * Setelah mengunduh, jalankan Ngrok dengan perintah:

bash

Copy

ngrok http 3000

* + Di sini, 3000 adalah port di mana aplikasi Next.js atau server pengembangan kamu berjalan (sesuaikan dengan port yang digunakan aplikasi kamu).

1. **Dapatkan URL Ngrok**:
   * Ngrok akan memberikan URL publik sementara seperti https://abcd1234.ngrok.io.
   * URL ini bisa kamu gunakan untuk mengonfigurasi **callback URL** di **Midtrans**.
2. **Konfigurasi Callback URL di Midtrans**:
   * Masuk ke **Midtrans Dashboard** dan tambahkan URL Ngrok sebagai **callback URL** untuk mengonfirmasi pembayaran.
   * Misalnya, jika aplikasi kamu berjalan di localhost:3000, Ngrok akan memberikan URL seperti https://abcd1234.ngrok.io.
   * Kamu bisa mengonfigurasi **Midtrans** untuk mengirimkan status transaksi ke URL ini.
3. **Pengujian Callback**:
   * Setelah mengonfigurasi, lakukan pengujian pembayaran melalui Midtrans. Midtrans akan mengirimkan data transaksi ke URL Ngrok, yang kemudian akan diteruskan ke aplikasi lokal kamu.

**Apakah Ngrok Diperlukan untuk Produksi?**

* **Tidak diperlukan** di lingkungan **produksi**. Ngrok hanya digunakan untuk **pengembangan lokal**.
* Di lingkungan produksi, kamu akan menggunakan **server yang memiliki URL publik** (misalnya, domain yang telah terdaftar atau aplikasi yang di-deploy di server atau cloud), dan **Midtrans akan mengirimkan callback** ke URL tersebut tanpa menggunakan Ngrok.

**Kapan Ngrok Tidak Diperlukan?**

* Jika aplikasi kamu sudah di-deploy di **server publik** (misalnya, menggunakan **Vercel**, **Netlify**, **AWS**, **Heroku**, atau lainnya), maka **callback URL** bisa langsung diatur ke URL yang ter-deploy tanpa memerlukan Ngrok.

**Kesimpulan:**

* **Ngrok dibutuhkan** saat pengembangan **lokal** jika kamu ingin menghubungkan aplikasi kamu dengan **Midtrans** untuk mengonfigurasi callback URL.
* **Ngrok tidak diperlukan** untuk aplikasi yang sudah di-deploy di **lingkungan produksi**, karena kamu akan menggunakan URL yang di-deploy untuk callback URL dari Midtrans.

Jika aplikasi kamu sudah siap untuk **produksi** dan ingin mengonfigurasi Midtrans di sana, **Ngrok tidak lagi diperlukan**. Namun, jika masih dalam tahap **pengembangan**, Ngrok akan memudahkan pengujian dan integrasi dengan Midtrans.

Apakah kamu ingin melanjutkan pengaturan ini untuk pengujian atau integrasi Midtrans lebih lanjut?

**You said:**

Saya masih berkutat pada Admin, apakah perlu langsung menghubungkan Midtrans lewat Ngrok pada payment, atau saya mengembangkan CRUD lain dulu seperti Ticket atau Review?

**ChatGPT said:**

Jika kamu masih dalam tahap **pengembangan Admin** dan sedang fokus pada pengelolaan data dan CRUD lainnya seperti **Ticket** atau **Review**, saya sarankan untuk **menyelesaikan pengembangan CRUD terlebih dahulu** sebelum mengintegrasikan Midtrans dengan **Ngrok**. Berikut alasan dan langkah yang bisa kamu ambil:

**Alasan Mengapa Menyelesaikan CRUD Terlebih Dahulu:**

1. **Fokus pada Fitur Inti Terlebih Dahulu**:
   * Sebelum menghubungkan Midtrans dan mengonfigurasi pembayaran, lebih baik memastikan bahwa fitur-fitur **CRUD dasar** seperti **Ticket** dan **Review** sudah berjalan dengan baik. Ini termasuk pengelolaan data untuk tiket yang akan dibeli dan ulasan yang diberikan oleh pengguna.
2. **Penyusunan Database dan Alur Pengguna yang Jelas**:
   * Dengan menyelesaikan CRUD untuk **Ticket** dan **Review**, kamu akan mempermudah pengelolaan data dan memastikan bahwa alur pengguna sudah berfungsi dengan baik sebelum pembayaran dan transaksi.
3. **Integrasi Pembayaran Setelah Fitur Utama Terbentuk**:
   * Integrasi dengan Midtrans untuk **pembayaran** lebih baik dilakukan setelah **proses pembelian tiket** dan **pengelolaan ulasan** siap, sehingga alur transaksi yang melibatkan pembayaran bisa berfungsi dengan baik dalam konteks aplikasi yang sudah stabil.

**Langkah-Langkah yang Disarankan:**

1. **Selesaikan CRUD untuk Ticket**:
   * Selesaikan pengelolaan **tiket** dan pastikan pengguna dapat membeli tiket untuk spot wisata tertentu. Kamu dapat membuat API CRUD untuk **ticket** dengan data seperti harga, stok, dan informasi lainnya.
   * Pastikan data transaksi tiket dapat disambungkan dengan **tabel transaction** dan **transaction\_detail**.
2. **Selesaikan CRUD untuk Review**:
   * Lanjutkan dengan membuat CRUD untuk **ulasan (Review)** agar pengguna dapat memberikan rating dan komentar untuk spot wisata yang mereka kunjungi.
   * Hubungkan **ulasan** dengan **spot wisata (spot\_tourism)** dan pastikan tabel **reviews** terisi dengan benar.
3. **Pastikan Alur Pengguna Berfungsi**:
   * Pastikan alur pengguna bekerja dengan baik dari mulai melihat daftar spot wisata, memilih tiket, memberikan ulasan, hingga akhirnya melakukan pembayaran.
4. **Setelah CRUD Stabil, Integrasikan Midtrans**:
   * Setelah kamu yakin bahwa CRUD dan alur transaksi sudah berjalan dengan baik, kamu bisa mulai mengintegrasikan **Midtrans** untuk menangani pembayaran.
   * Gunakan **Ngrok** untuk menghubungkan aplikasi lokal dengan Midtrans jika masih dalam tahap pengembangan.

**Keuntungan dari Pendekatan Ini:**

* **Meminimalkan Gangguan**: Dengan menyelesaikan pengelolaan data terlebih dahulu, kamu bisa fokus pada bagian pembayaran tanpa khawatir dengan masalah pengelolaan data yang bisa mempengaruhi alur transaksi.
* **Pengujian Lebih Rapi**: Setelah fitur-fitur inti berfungsi, kamu bisa melakukan pengujian pembayaran (melalui Midtrans) dengan data yang sudah stabil dan siap pakai.

**Tindakan yang Dapat Dilakukan Sekarang:**

* Fokuskan pada CRUD **Ticket** dan **Review** terlebih dahulu.
* Pastikan tabel ticket, transaction, dan transaction\_detail sudah bekerja dengan baik.
* Setelah itu, kamu bisa kembali ke bagian **Payment Integration** dengan Midtrans dan **menggunakan Ngrok** untuk menghubungkan aplikasi lokal dengan Midtrans API.

Jika kamu siap untuk lanjut dengan pengembangan **Ticket** atau **Review**, beri tahu saya dan kita bisa mulai membahasnya lebih lanjut!

**You said:**

Baik saya akan fokus ke CRUD Ticket dulu pada chat terbaru. Nantinya chat ini "Lanjutan Implementasi CRUD Desa (Gallery)" akan berkutat dengan Midtrans.

**ChatGPT said:**

Updated saved memory

Siap! Fokus pada **CRUD Ticket** terlebih dahulu adalah langkah yang tepat, dan nanti kita bisa kembali ke integrasi **Midtrans** setelah semua fitur dasar seperti ticketing dan pembayaran berjalan dengan baik.

Jika ada kode yang ingin kamu diskusikan atau perlu bantuan, kirimkan saja, dan saya siap membantu! 🚀