Admin CRUD Lanjutan.docx

Document

Saya ingin melanjutkan chat sebelumnya yaitu "Admin CRUD Lanjutan" dalam folder proyek "Jmko - NEXTJS + MYSQL + PRISMA + MIDTRANS, chat Admin CRUD dengan Midtrans". Untuk spot\_tourism (menjadi folder destination pada arsitektur folder) dan ticket (folder ticket) sudah saya buatkan API CRUD-nya. Saat ini saya sedikit kesulitan dalam API untuk Payment, mengingat data-data dalam tabel database memang lebih terisi.

**ChatGPT said:**

Tentu! Berikut adalah langkah-langkah yang perlu kamu ikuti untuk mengatasi masalah dan melanjutkan proses input data:

1. **Periksa Tabel payment\_method**  
   Pastikan bahwa method\_id yang kamu coba masukkan ke tabel payment sudah ada di tabel payment\_method. Jika belum ada, kamu perlu menambahkannya terlebih dahulu. Misalnya, untuk QRIS:

sql

CopyEdit

INSERT INTO payment\_method (method\_id, provider\_id, method\_name, method\_code, method\_icon, service\_fee, minimum\_payment, maximum\_payment, is\_active)

VALUES (1, 'QRIS', 'QRIS', 'QRIS01', 'https://linktoicon.com/qr-icon.png', 5000, 10000, 1000000, 1);

1. **Periksa Tabel payment\_provider**  
   Sebelum kamu memasukkan data ke tabel payment\_method, pastikan bahwa provider\_id seperti 'QRIS' sudah ada di tabel payment\_provider. Kamu bisa menggunakan query ini untuk memastikan:

sql

CopyEdit

SELECT \* FROM payment\_provider WHERE provider\_id = 'QRIS';

Jika tidak ada, kamu perlu menambahkannya terlebih dahulu:

sql

CopyEdit

INSERT INTO payment\_provider (provider\_id, provider\_name, provider\_desc, provider\_logo, is\_active)

VALUES ('QRIS', 'QRIS', 'QRIS is a payment method using QR codes.', 'https://linktoicon.com/qris-logo.png', 1);

1. **Masukkan Data ke Tabel payment**  
   Setelah memastikan data di payment\_method dan payment\_provider sudah ada, kamu bisa lanjut memasukkan data ke tabel payment. Pastikan method\_id yang digunakan sudah sesuai, misalnya 1 untuk QRIS:

sql

CopyEdit

INSERT INTO payment (transaction\_id, method\_id, reference\_number, payment\_amount, service\_fee, total\_payment, payment\_status, payment\_time, expiration\_time)

VALUES (3, 1, 'REF-123456789', 100000, 5000, 105000, 'completed', NOW(), DATE\_ADD(NOW(), INTERVAL 1 HOUR));

1. **Verifikasi Data**  
   Setelah data berhasil dimasukkan, kamu bisa memverifikasi dengan query:

sql

CopyEdit

SELECT \* FROM payment;

Jika kamu mengikuti langkah-langkah ini dan memasukkan data dengan benar, seharusnya masalah foreign key constraints dapat teratasi.

Apakah sudah berhasil? Jika masih ada masalah atau pertanyaan, beri tahu saya, saya siap bantu! 😊

**You said:**

Mungkin saya lewat dulu, saya ingin membangun API reviews.

**ChatGPT said:**

Siap! 🔥  
Kita langsung fokus ke **membangun API Reviews** ya.

Untuk API Admin bagian **reviews**, tadi rencananya:

| **Fitur** | **Aksi** | **Keterangan** |
| --- | --- | --- |
| GET Reviews | Melihat semua review | Untuk menampilkan semua ulasan yang masuk |
| DELETE Review | Menghapus review tertentu | Jika ada review tidak pantas / spam |

**📦 Struktur Folder API Reviews**

Kita buat di:

pgsql

CopyEdit

/pages

/api

/admin

/reviews

route.ts --> untuk GET semua review

delete/[id].ts --> untuk DELETE review berdasarkan review\_id

**✍️ 1. GET Semua Reviews**

File: /pages/api/admin/reviews/route.ts

typescript

CopyEdit

// pages/api/admin/reviews/route.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@/lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

if (req.method === 'GET') {

try {

const reviews = await prisma.review.findMany({

orderBy: {

created\_at: 'desc', // urutkan review terbaru

},

include: {

user: true, // kalau mau ambil data user yang memberikan review

spot\_tourism: true, // kalau mau ambil data destinasi yang direview

},

});

res.status(200).json(reviews);

} catch (error) {

console.error('Error fetching reviews:', error);

res.status(500).json({ error: 'Failed to fetch reviews' });

}

} else {

res.status(405).json({ error: 'Method Not Allowed' });

}

}

**✍️ 2. DELETE Review by ID**

File: /pages/api/admin/reviews/delete/[id].ts

typescript

CopyEdit

// pages/api/admin/reviews/delete/[id].ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@/lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

const { id } = req.query; // review\_id

if (req.method === 'DELETE') {

try {

await prisma.review.delete({

where: {

review\_id: Number(id),

},

});

res.status(200).json({ message: 'Review deleted successfully' });

} catch (error) {

console.error('Error deleting review:', error);

res.status(500).json({ error: 'Failed to delete review' });

}

} else {

res.status(405).json({ error: 'Method Not Allowed' });

}

}

**🚀 Cara Testing di Postman**

| **Aksi** | **Method** | **URL** |
| --- | --- | --- |
| Lihat Semua Reviews | GET | http://localhost:3000/api/admin/reviews/route |
| Hapus Review | DELETE | http://localhost:3000/api/admin/reviews/delete/{review\_id} |

Contoh DELETE:

* Method: DELETE
* URL: http://localhost:3000/api/admin/reviews/delete/5 (ganti 5 dengan review\_id yang ada)

**🎯 Recap API Reviews Admin**

| **Fitur** | **Status** |
| --- | --- |
| GET | ✔️ |
| DELETE | ✔️ |

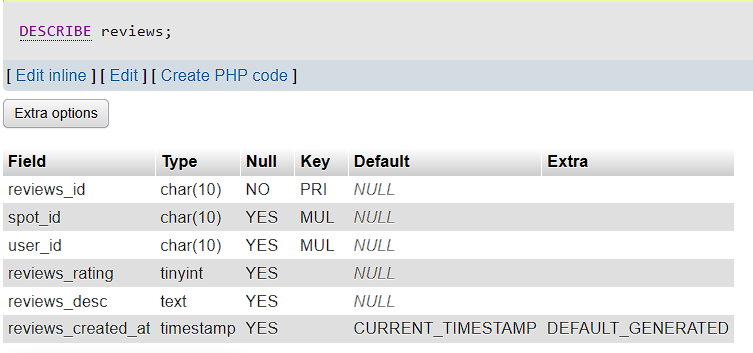
Kalau kamu setuju dengan struktur dan kodingan di atas, kamu bisa langsung:

* Copy kode GET
* Copy kode DELETE
* Testing via Postman

Kalau mau, aku bisa lanjut sekalian buatin **dummy data** untuk review biar kamu langsung bisa testing GET dan DELETE. ✨

Mau lanjut testing langsung atau mau sekalian aku bantu buat dummy review dulu? 🚀  
Kalau mau, tinggal bilang "Lanjut testing" atau "Buat dummy review dulu".

**You said:**



Saya sudah memasukan route.ts untuk GET pada review, tapi terjadi eror Property 'review' does not exist on type 'PrismaClient<PrismaClientOptions, never, DefaultArgs>'. Did you mean 'reviews'? // pages/api/admin/reviews/route.ts import { NextApiRequest, NextApiResponse } from 'next'; import prisma from '@/lib/prisma'; export default async function handler(req: NextApiRequest, res: NextApiResponse) { if (req.method === 'GET') { try { const reviews = await prisma.review.findMany({ orderBy: { created\_at: 'desc', // urutkan review terbaru }, include: { user: true, // kalau mau ambil data user yang memberikan review spot\_tourism: true, // kalau mau ambil data destinasi yang direview }, }); res.status(200).json(reviews); } catch (error) { console.error('Error fetching reviews:', error); res.status(500).json({ error: 'Failed to fetch reviews' }); } } else { res.status(405).json({ error: 'Method Not Allowed' }); } } ============================ Dan ini struktur tabel di reviews yang dari saya screenshot.

**ChatGPT said:**

Terima kasih atas screenshot DESCRIBE reviews—ini sangat membantu!  
Dari error dan struktur tabel, **penyebab error-mu sudah jelas**:

**🧨 Penyebab Error**

Error:

ts

CopyEdit

Property 'review' does not exist on type 'PrismaClient'

Ini terjadi karena nama model di schema.prisma kamu adalah reviews (pakai s), **bukan** review.

**✅ Solusi**

Ubah:

ts

CopyEdit

prisma.review.findMany(...)

Menjadi:

ts

CopyEdit

prisma.reviews.findMany(...)

Jadi kode lengkapnya menjadi:

ts

CopyEdit

// pages/api/admin/reviews/route.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@/lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

if (req.method === 'GET') {

try {

const reviews = await prisma.reviews.findMany({

orderBy: {

reviews\_created\_at: 'desc', // nama kolom timestamp sesuai screenshot

},

include: {

user: true, // pastikan relasi ini ada di schema.prisma

spot\_tourism: true, // pastikan relasi ini juga ada

},

});

res.status(200).json(reviews);

} catch (error) {

console.error('Error fetching reviews:', error);

res.status(500).json({ error: 'Failed to fetch reviews' });

}

} else {

res.status(405).json({ error: 'Method Not Allowed' });

}

}

**🧩 Bonus: Cek Schema**

Pastikan di file schema.prisma kamu, modelnya memang seperti ini:

prisma

CopyEdit

model reviews {

reviews\_id String @id

spot\_id String? @db.Char(10)

user\_id String? @db.Char(10)

reviews\_rating Int?

reviews\_desc String?

reviews\_created\_at DateTime? @default(now())

user users? @relation(fields: [user\_id], references: [user\_id])

spot\_tourism spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id])

}

Kalau relasi user atau spot\_tourism tidak ada, kamu bisa hapus bagian include dulu.

**🚀 Langkah Selanjutnya:**

1. Perbaiki nama model review ➝ reviews.
2. Pastikan field reviews\_created\_at ditulis dengan nama yang sesuai.
3. Kalau relasi belum di-define di schema.prisma, hapus dulu include sementara waktu.
4. Jalankan ulang GET di Postman.

Kalau sudah berhasil, tinggal bilang:  
**"GET reviews berhasil!"**  
Atau kalau error baru muncul, kirim lagi dan aku bantu debug 💪

**You said:**

Ini model untuk reviews di Prisma 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") }

**ChatGPT said:**

Mantap! Dari model reviews yang kamu kirim, kita bisa perbaiki kode GET API-nya 100% akurat. 🔍  
Berikut ini **hal-hal penting yang perlu diperhatikan** dari modelmu:

**🧠 Ringkasan Model:**

| **Field** | **Tipe** | **Keterangan** |
| --- | --- | --- |
| reviews\_id | @id (primary key) | ✅ OK |
| spot | relasi ke spot\_tourism via spot\_id | ✅ OK |
| user | relasi ke users via user\_id | ✅ OK |
| Nama model | reviews | ✅ Pakai prisma.reviews |

**✅ Perbaikan Kode GET Reviews (Final Fix)**

ts

CopyEdit

// pages/api/admin/reviews/route.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@/lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

if (req.method === 'GET') {

try {

const reviews = await prisma.reviews.findMany({

orderBy: {

reviews\_created\_at: 'desc',

},

include: {

user: true, // dari relasi users?

spot: true, // dari relasi spot\_tourism?

},

});

res.status(200).json(reviews);

} catch (error) {

console.error('Error fetching reviews:', error);

res.status(500).json({ error: 'Failed to fetch reviews' });

}

} else {

res.status(405).json({ error: 'Method Not Allowed' });

}

}

**🔁 Penyesuaian:**

* prisma.reviews.findMany() ✅
* include: { user, spot } ✅ sesuai relasi yang kamu definisikan
* orderBy: reviews\_created\_at ✅ sesuai kolom timestamp

**🚀 Langkah Lanjutan:**

1. Copy-paste kode di atas ke file route.ts untuk /api/admin/reviews.
2. Jalankan ulang GET dari Postman ke:

bash

CopyEdit

http://localhost:3000/api/admin/reviews/route

1. Pastikan ada data review di database agar hasil tidak kosong.

Kalau kamu sudah testing dan data muncul, tinggal balas:

✅ **"GET Reviews berhasil!"**

Kalau mau lanjut ke DELETE atau butuh dummy data untuk test, tinggal bilang aja! 🎯

**You said:**

Mungkin saya akan lewatkan API reviews dulu dalam pengecekan.

**ChatGPT said:**

Oke, siap! ✨  
Kalau kamu mau **lewatkan dulu** pengecekan API reviews, itu **nggak masalah sama sekali**.

Kita tetap bisa lanjut ke bagian lain tanpa terganggu.

**🚀 Recap Status Sekarang:**

| **Bagian** | **Status** |
| --- | --- |
| Transaction (GET + PUT) | ✅ Sudah selesai |
| Payment (GET) | ✅ Sudah selesai |
| Reviews (GET + DELETE) | ➡️ Sudah dibuat kodenya, pengecekan bisa nanti |

**📚 Opsi Lanjut:**

Kalau mau lanjut progress proyek admin kita, ada dua pilihan:

* **Lanjut API CRUD lain** (seperti: category\_tourism, facility, gallery, dll — opsional).
* **Mulai bangun Frontend Admin Panel** pakai **shadcn/ui** 🎨 (untuk tampilkan daftar transaksi, pembayaran, dll).

✋ Tinggal pilih:

* Kalau mau lanjut API CRUD ➔ balas: "Lanjut CRUD"
* Kalau mau mulai Frontend Admin ➔ balas: "Mulai Frontend Admin"

Aku siap lanjut bantuin jalan cepat sesuai yang kamu mau! 🚀  
Mau pilih yang mana? 🎯

**You said:**

Saya ingin lanjut ke desain Admin Panel FE menggunakan shadcn/ui.

**ChatGPT said:**

Wuihhh, mantap! 🔥✨  
Mulai **desain Admin Panel Frontend** pakai **shadcn/ui** sekarang adalah pilihan **super tepat** setelah backend API dasar beres.

Aku langsung siapin langkah awalnya biar rapi:

**🚀 Roadmap Membangun Admin Panel FE (dengan shadcn/ui)**

| **Tahap** | **Yang Dibuat** | **Keterangan** |
| --- | --- | --- |
| 1 | Setup struktur halaman Admin | Buat /admin dashboard |
| 2 | Halaman List Transaction | Tarik API /admin/transaction ➔ tampilkan table |
| 3 | Halaman List Payment | Tarik API /admin/payment ➔ tampilkan table |
| 4 | Halaman List Reviews (optional) | Kalau mau nanti, dari /admin/reviews |
| 5 | Update Status Transaksi | Tambah fitur **Edit Status** dari Table |
| 6 | Desain Responsive UI | Pastikan bisa diakses dari laptop / hp admin |

**📦 Struktur Folder Frontend**

Folder kamu akan seperti ini:

bash

CopyEdit

/src

/app

/admin

/dashboard (Landing admin)

/transaction (List semua transaksi)

/payment (List semua pembayaran)

/reviews (List semua review) [optional]

/components

/admin

TransactionTable.tsx

PaymentTable.tsx

ReviewTable.tsx

Kita pakai shadcn/ui untuk komponen:

* **Data Table** untuk daftar transaksi, pembayaran, review.
* **Dialog** atau **Dropdown** untuk update status transaksi.
* **Badge** untuk label status: pending, paid, cancelled.

**✨ Mulai Step 1: Setup Halaman Admin**

Pertama kita buat halaman landing /admin/dashboard sederhana dulu.

**1. Buat file baru:**

bash

CopyEdit

src/app/admin/page.tsx

Kalau kamu mau dashboard landingnya, isi begini:

tsx

CopyEdit

// src/app/admin/page.tsx

import { Card, CardContent } from "@/components/ui/card";

export default function AdminDashboardPage() {

return (

<div className="p-6">

<h1 className="text-3xl font-bold mb-6">Admin Dashboard</h1>

<div className="grid grid-cols-1 md:grid-cols-3 gap-6">

<Card>

<CardContent className="p-6">

<p className="text-xl font-semibold">Transactions</p>

<p className="text-gray-500">View and manage transactions</p>

</CardContent>

</Card>

<Card>

<CardContent className="p-6">

<p className="text-xl font-semibold">Payments</p>

<p className="text-gray-500">View payment records</p>

</CardContent>

</Card>

<Card>

<CardContent className="p-6">

<p className="text-xl font-semibold">Reviews</p>

<p className="text-gray-500">Manage user reviews</p>

</CardContent>

</Card>

</div>

</div>

);

}

**📣 Hasil:**

* Dashboard admin pertama akan tampil 3 buah kartu: **Transactions**, **Payments**, **Reviews**.

**📋 Checklist Step Ini:**

✅ Buat /src/app/admin/page.tsx.

✅ Pakai Card dari shadcn/ui untuk tampilan dashboard.

✅ Pastikan route http://localhost:3000/admin menampilkan dashboard.

**🚀 Next Step**

Kalau sudah beres dashboard:

➡️ lanjut Step 2 ➔ Mulai buat halaman **Transaction List** (/admin/transaction), narik data dari API GET /api/admin/transaction/route.

**✋ Mau langsung lanjut ke Step 2 (Transaction Table)?**

Kalau mau tinggal balas:

**"Lanjut ke Transaction Table"**

Kalau mau aku bantu sekalian buatin Sidebar Admin Layout + Routing sekalian (biar dashboard admin kamu keren & rapi), tinggal bilang:

**"Buatkan Sidebar Admin sekalian"** 🎨

Mau pilih yang mana dulu? 🚀✨

**You said:**

Mungkin saya akan membuat chat baru lagi dari kelanjutan chat ini agar tidak terlalu panjang dan khusus untuk desain panel Admin menggunakan shacdn/ui.

**ChatGPT said:**

Siap banget! 💯  
Itu keputusan yang **rapi dan strategis** — biar pembahasan panel admin pakai **shadcn/ui** fokus dan nggak tercampur dengan backend sebelumnya.

Kalau kamu nanti buka chat baru, kamu bisa langsung awali dengan kalimat seperti ini:

“Saya ingin melanjutkan desain Admin Panel menggunakan shadcn/ui. Di chat sebelumnya saya sudah menyelesaikan API transaction dan payment, dan sudah membuat halaman /admin/page.tsx.”

Lalu tinggal bilang:

“Sekarang saya ingin mulai halaman list transaksi / table transaksi.”

Aku langsung akan sambut dan bantuin lanjut progress dengan struktur, komponen, dan implementasi frontend admin sebersih dan seresponsif mungkin! 🚀

Kalau nanti mau lanjut, tinggal mulai chat baru ya — aku tunggu di sana! ✨  
Sukses terus! 💻🔥