Berikut adalah **ringkasan alur kerja Next.js untuk menjalankan CRUD** dengan Prisma dan MySQL:

## 1. Setup Project

**Buat folder proyek** dan inisialisasi Next.js:

|  |
| --- |
| npx create-next-app@latest next\_crud  cd next\_crud |

**Instalasi Prisma & MySQL Client**:

|  |
| --- |
| npm install @prisma/client @next-auth/prisma-adapter mysql  npx prisma init |

**Konfigurasi .env** dengan MySQL:

|  |
| --- |
| DATABASE\_URL="mysql://root:@localhost:3306/next\_crud"  DATABASE\_URL="mysql://root:@localhost:3306/nama\_database"(template) |

## 2. Konfigurasi Prisma & Database

**Edit prisma/schema.prisma**:

|  |
| --- |
| generator client {  provider = "prisma-client-js"  }  datasource db {  provider = "mysql"  url = env("DATABASE\_URL")  }  model User {  id Int @id @default(autoincrement())  name String  email String @unique  createdAt DateTime @default(now())  } |

**Jalankan migrasi database**:

|  |
| --- |
| npx prisma migrate dev --name init |

## 3. Buat API Routes di Next.js

**Buat file app/api/users/route.js**:

|  |
| --- |
| import { NextResponse } from "next/server";  import { PrismaClient } from "@prisma/client";  const prisma = new PrismaClient();  // GET: Ambil semua data user  export async function GET() {  const users = await prisma.user.findMany();  return NextResponse.json(users);  }  // POST: Tambah user baru  export async function POST(req) {  const { name, email } = await req.json();  try {  const user = await prisma.user.create({ data: { name, email } });  return NextResponse.json(user, { status: 201 });  } catch (error) {  return NextResponse.json({ error: "Gagal menambah user" }, { status: 500 });  }  }  // PUT: Update user  export async function PUT(req) {  const { id, name, email } = await req.json();  try {  const user = await prisma.user.update({ where: { id }, data: { name, email } });  return NextResponse.json(user);  } catch (error) {  return NextResponse.json({ error: "Gagal mengupdate user" }, { status: 500 });  }  }  // DELETE: Hapus user  export async function DELETE(req) {  const { id } = await req.json();  try {  await prisma.user.delete({ where: { id } });  return NextResponse.json({ message: "User berhasil dihapus" });  } catch (error) {  return NextResponse.json({ error: "Gagal menghapus user" }, { status: 500 });  }  } |

## 4. Buat Frontend di app/page.js

**Buat file app/page.js**:

|  |
| --- |
| "use client";  import { useState, useEffect } from "react";  export default function Home() {  const [users, setUsers] = useState([]);  const [name, setName] = useState("");  const [email, setEmail] = useState("");  const [editingUser, setEditingUser] = useState(null);  useEffect(() => {  fetchUsers();  }, []);  const fetchUsers = async () => {  const res = await fetch("/api/users");  const data = await res.json();  setUsers(data);  };  const addUser = async (e) => {  e.preventDefault();  await fetch("/api/users", {  method: "POST",  headers: { "Content-Type": "application/json" },  body: JSON.stringify({ name, email }),  });  setName("");  setEmail("");  fetchUsers();  };  const updateUser = async (e) => {  e.preventDefault();  await fetch("/api/users", {  method: "PUT",  headers: { "Content-Type": "application/json" },  body: JSON.stringify({ id: editingUser.id, name, email }),  });  setEditingUser(null);  setName("");  setEmail("");  fetchUsers();  };  const deleteUser = async (id) => {  await fetch("/api/users", {  method: "DELETE",  headers: { "Content-Type": "application/json" },  body: JSON.stringify({ id }),  });  fetchUsers();  };  return (  <div style={{ padding: "20px", maxWidth: "500px", margin: "auto" }}>  <h2>Daftar Pengguna</h2>  <ul>  {users.map((user) => (  <li key={user.id}>  {user.name} - {user.email}  <button onClick={() => deleteUser(user.id)} style={{ marginLeft: "10px" }}>Hapus</button>  <button onClick={() => { setEditingUser(user); setName(user.name); setEmail(user.email); }}>Edit</button>  </li>  ))}  </ul>  <h3>{editingUser ? "Edit Pengguna" : "Tambah Pengguna"}</h3>  <form onSubmit={editingUser ? updateUser : addUser}>  <input type="text" placeholder="Nama" value={name} onChange={(e) => setName(e.target.value)} required />  <input type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required />  <button type="submit">{editingUser ? "Update" : "Tambah"}</button>  </form>  </div>  );  } |

## 5. Jalankan Project

**Jalankan server Next.js**:

|  |
| --- |
| npm run dev |

**Tes API di Postman atau browser**:

* **GET** → http://localhost:3000/api/users
* **POST** → Tambah user
* **PUT** → Update user
* **DELETE** → Hapus user

**Akses website**:

* **Buka** http://localhost:3000 di browser.

## Kesimpulan

 **Backend (API)** dibuat dengan **Next.js API Routes** dan **Prisma** untuk CRUD database.

 **Frontend** menggunakan **React Hooks** (useState, useEffect) untuk menampilkan dan mengelola data.

 **Database** menggunakan **MySQL dengan Prisma ORM**.

 **Semua fitur CRUD (Create, Read, Update, Delete)** telah berjalan dengan baik.