Kepada Yth. Tim Rekrutmen PT. XYZ,

Berikut adalah jawaban saya untuk tes asesmen *Application Design* yang diberikan. Saya telah berusaha menyusun jawaban ini secara komprehensif dan terstruktur sesuai dengan tugas yang diminta.

**Analisis Kebutuhan (Berdasarkan User Story)**

Sebelum masuk ke desain teknis, saya merangkum kebutuhan fungsional dari *user story* yang diberikan:

1. Manajemen Pengguna: Registrasi (data diri, email, telepon, KTP, foto diri) dan Login (password/biometrik).
2. Manajemen Pinjaman: Pengajuan pinjaman baru dengan batasan (maks Rp 12.000.000, tenor 1 tahun), proses persetujuan/penolakan, dan validasi untuk mencegah pinjaman ganda (jika ada yang masih aktif atau dalam proses).
3. Dasbor Informasi: Pengguna dapat melihat sisa hutang dan detail tagihan bulanan.
4. Sistem Notifikasi: Pemberitahuan status pinjaman (diterima/ditolak) melalui email dan nomor telepon.

Berdasarkan analisis ini, saya menyusun desain berikut:

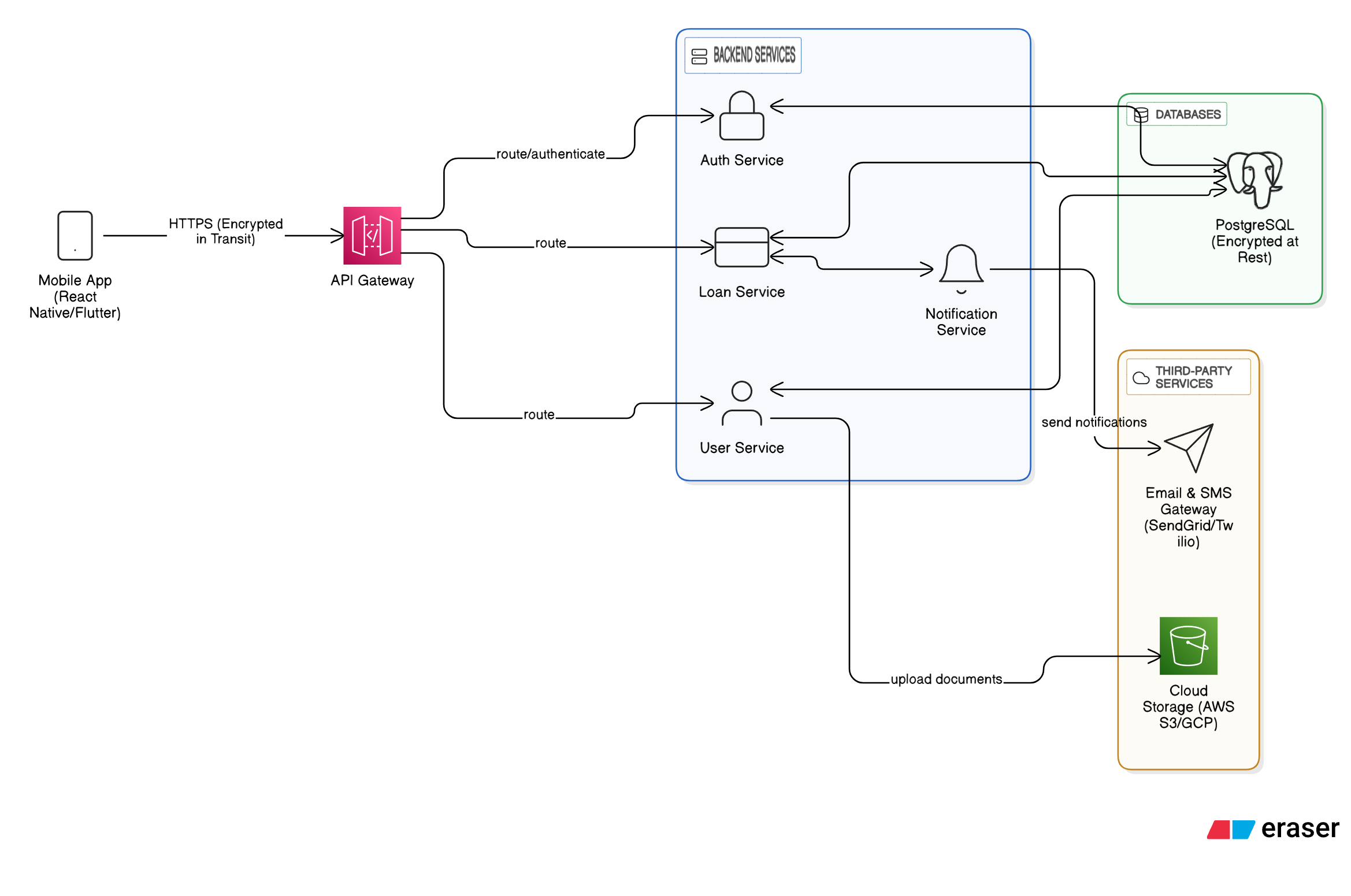
**Tugas 1: High Level Design Architecture**

Arsitektur sistem ini akan dirancang dengan pendekatan *microservices* untuk memastikan skalabilitas, kemudahan pemeliharaan, dan fleksibilitas di masa depan.

Komponen Utama:

1. Mobile Application (Client): Aplikasi yang akan digunakan oleh user. Dibangun menggunakan teknologi cross-platform seperti React Native atau Flutter untuk efisiensi pengembangan di platform iOS dan Android.
2. API Gateway: Pintu gerbang utama untuk semua permintaan dari aplikasi mobile. Bertugas untuk *routing*, *rate limiting*, dan otentikasi awal.
3. Backend Services (Microservices):
   * Auth Service: Mengelola semua hal terkait otentikasi dan otorisasi, seperti registrasi, login, dan manajemen token (JWT).
   * User Service: Mengelola data profil pengguna, termasuk verifikasi data (e-KYC).
   * Loan Service: Mengelola seluruh logika bisnis terkait pinjaman, mulai dari pengajuan, kalkulasi, persetujuan, hingga pelunasan.
   * Notification Service: Bertugas mengirimkan notifikasi email dan SMS/WhatsApp kepada pengguna.
4. Database:
   * PostgreSQL (Relational Database): Digunakan untuk menyimpan data transaksional yang terstruktur seperti data user, pinjaman, dan jadwal pembayaran.
   * Redis (In-memory Database): Digunakan untuk *caching* dan manajemen sesi/token agar performa lebih cepat.
5. Third-Party Services:
   * Cloud Storage (e.g., AWS S3 atau Google Cloud Storage): Untuk menyimpan file statis seperti foto KTP dan foto diri pengguna secara aman.
   * Email & SMS Gateway (e.g., SendGrid, Twilio): Untuk mengirimkan notifikasi.
   * e-KYC & Credit Scoring Service: Pihak ketiga yang terintegrasi untuk memverifikasi identitas pengguna (Know Your Customer) dan menilai kelayakan kredit mereka. Ini adalah komponen krusial dalam bisnis pinjaman online.

Diagram Arsitektur:



**Tugas 2: Spesifikasi Design Screen Flow dan ERD**

**A. Screen Flow (Alur Layar Aplikasi)**

Berikut adalah alur dasar yang akan dilalui pengguna:

1. Alur Pendaftaran & Login:  
   Splash Screen -> Halaman Selamat Datang (Pilihan Login / Daftar)
   * Daftar: Input Email & Password -> Input Data Diri & No. Telepon -> Upload Foto KTP & Foto Diri -> Verifikasi OTP (via SMS) -> Pendaftaran Selesai (Menunggu Verifikasi Data) -> Login
   * Login: Input Email & Password -> (Opsional) Otentikasi Biometrik -> Dasbor Utama
2. Alur Pengajuan Pinjaman:  
   Dasbor Utama -> Tombol "Ajukan Pinjaman" -> Halaman Kalkulator Pinjaman (Input Jumlah & Tenor) -> Halaman Rincian & Konfirmasi Pinjaman -> Input PIN/Biometrik Konfirmasi -> Halaman "Pengajuan Terkirim" -> Kembali ke Dasbor Utama (dengan status "Dalam Proses")
3. Alur Melihat Tagihan:  
   Dasbor Utama -> Lihat Detail Pinjaman -> Halaman Rincian Pinjaman Aktif (Sisa Hutang, Jatuh Tempo Berikutnya, Riwayat Pembayaran)

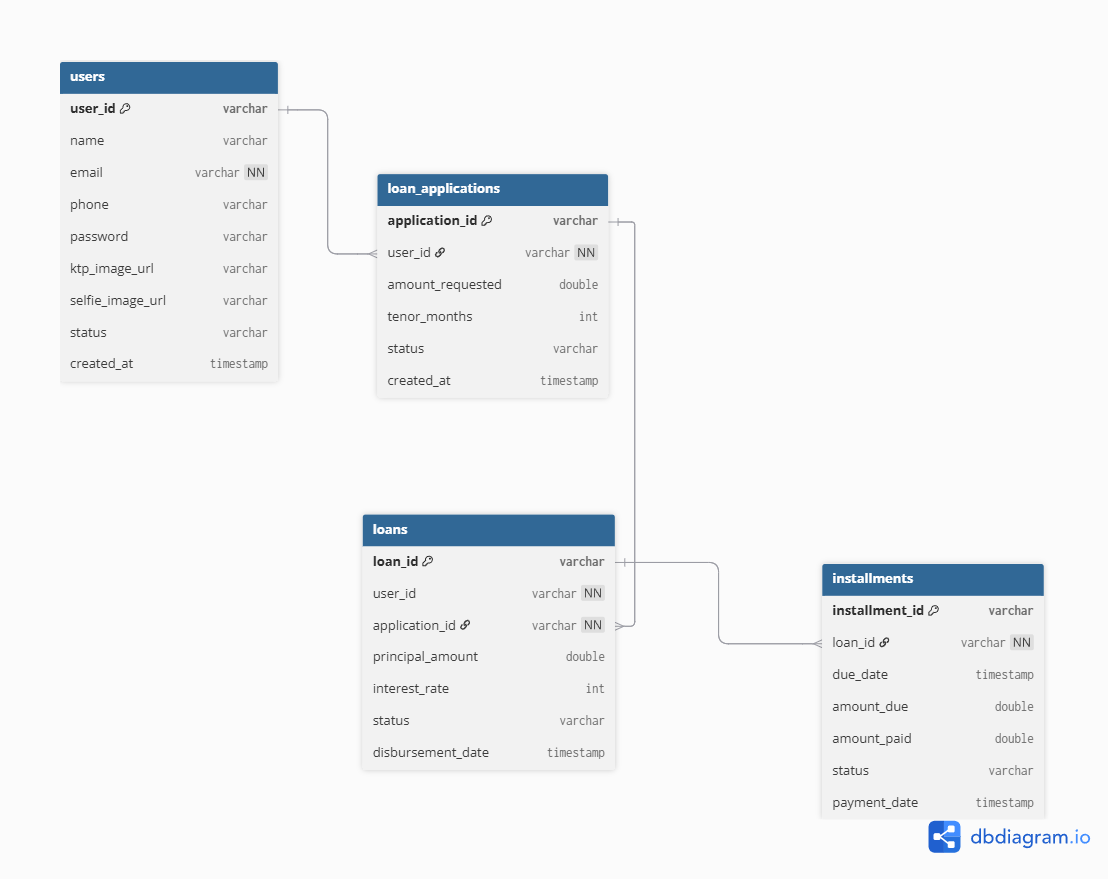
**B. Entity-Relationship Diagram (ERD)**

ERD ini menggambarkan struktur tabel utama dalam database PostgreSQL.

Entitas:

* users: Menyimpan data pengguna.
* loan\_applications: Menyimpan riwayat setiap pengajuan pinjaman.
* loans: Menyimpan data pinjaman yang telah disetujui.
* installments: Menyimpan rincian tagihan per bulan untuk setiap pinjaman.

Diagram:



*Relasi:*

* Satu user bisa memiliki banyak loan\_applications.
* Satu loan\_application yang disetujui akan menghasilkan satu loan.
* Satu loan memiliki banyak installments (tagihan bulanan).

**Tugas 3: Detail Design untuk API (UML, Flowchart)**

Saya akan mencontohkan desain API untuk proses Pengajuan Pinjaman menggunakan format REST API dan digambarkan dengan *Sequence Diagram* (salah satu bentuk UML).

**API Endpoint: POST /api/v1/loans/apply**

* Deskripsi: Endpoint untuk user mengajukan pinjaman baru.
* Method: POST
* Authorization: Bearer Token (JWT yang didapat saat login).
* Request Body (JSON):

{

"amount": 1000000, // Jumlah pinjaman dalam Rupiah

"tenor": 6 // Tenor dalam bulan

}

* Success Response (201 Created):

{

"application\_id": "app\_123xyz",

"status": "pending",

"message": "Loan application submitted successfully and is under review."

}

Error Response (400 Bad Request / 409 Conflict):

* Jika user masih punya pinjaman aktif:

{

"error": "Conflict",

"message": "You already have an active or pending loan."

}

* Jika input tidak valid (misal, amount > 12jt):

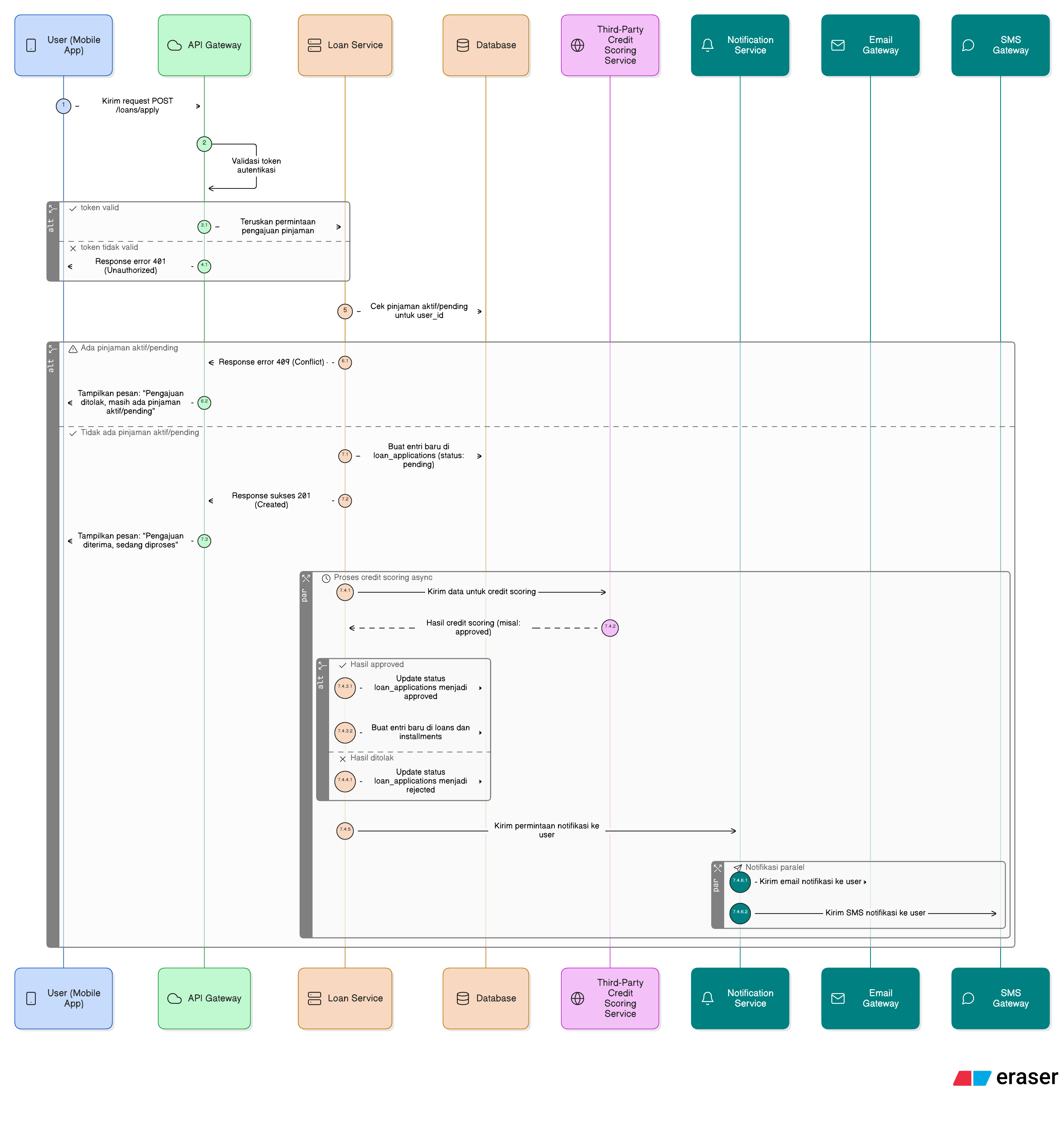
{

"error": "Bad Request",

"message": "Loan amount exceeds the maximum limit of Rp 12,000,000."

}

**Sequence Diagram: Alur Proses Pengajuan Pinjaman**

**

*Deskripsi Diagram:*

1. User (Mobile App) mengirim request POST /loans/apply ke API Gateway.
2. API Gateway memvalidasi token dan meneruskan request ke Loan Service.
3. Loan Service melakukan validasi bisnis:
   * Ia memeriksa ke database (loans table) apakah user\_id tersebut memiliki pinjaman dengan status active atau pending.
   * Jika ada, ia akan mengembalikan response error 409 (Conflict).
4. Jika tidak ada pinjaman aktif, Loan Service akan membuat entri baru di tabel loan\_applications dengan status pending.
5. Loan Service mengembalikan response sukses 201 (Created) ke user.
6. (Proses Asynchronous) Secara terpisah, Loan Service memulai proses *credit scoring* dengan memanggil Third-Party Credit Scoring Service.
7. Setelah mendapatkan hasil (misal, approved), Loan Service akan:
   * Update status di loan\_applications menjadi approved.
   * Membuat entri baru di tabel loans dan installments.
8. Terakhir, Loan Service memanggil Notification Service untuk memberitahu user.
9. Notification Service mengirim notifikasi via Email Gateway dan SMS Gateway.

**Tugas 4: Detail Design untuk Screen Behavior**

Berikut adalah detail perilaku untuk beberapa layar utama berdasarkan *screen flow* yang telah dibuat.

**1. Screen: Dasbor Utama (Home Screen)**

* State / Kondisi:
  + Kondisi 1: Tidak Ada Pinjaman Aktif.
    - UI: Menampilkan pesan sambutan, "Selamat datang, [Nama User]!", dan sebuah kartu besar dengan tulisan "Anda belum memiliki pinjaman aktif. Ajukan sekarang!" beserta tombol "Ajukan Pinjaman" yang aktif.
    - Behavior: Saat layar dimuat (onLoad), aplikasi memanggil API GET /api/v1/users/me/loans. Jika response kosong, UI akan ditampilkan sesuai kondisi ini. Tombol "Ajukan Pinjaman" akan mengarahkan user ke layar kalkulator pinjaman.
  + Kondisi 2: Ada Pinjaman Aktif.
    - UI: Menampilkan kartu ringkasan pinjaman yang berisi: "Sisa Hutang: Rp 8.000.000", "Tagihan Berikutnya: Rp 1.500.000 jatuh tempo pada 25 September 2025". Tombol "Ajukan Pinjaman" akan dinonaktifkan (*disabled*) atau disembunyikan. Terdapat tombol "Lihat Rincian".
    - Behavior: onLoad, API GET /api/v1/users/me/loans mengembalikan data pinjaman aktif. Data ini digunakan untuk mengisi kartu ringkasan. Tombol "Ajukan Pinjaman" yang nonaktif akan menampilkan *tooltip* "Anda hanya dapat mengajukan setelah pinjaman saat ini lunas." jika diklik.
  + Kondisi 3: Pengajuan Dalam Proses.
    - UI: Menampilkan kartu dengan status: "Pengajuan Pinjaman Anda Sebesar Rp 5.000.000 sedang dalam proses review." Tombol "Ajukan Pinjaman" juga nonaktif.
    - Behavior: Sama seperti kondisi 2, status ini didapat dari API.

**2. Screen: Kalkulator Pinjaman (Loan Application Screen)**

* UI Elements:
  + Slider atau input field untuk "Jumlah Pinjaman", dengan rentang (misal, Rp 500.000 - Rp 12.000.000).
  + Pilihan (misal, buttons atau dropdown) untuk "Tenor Pinjaman" (misal, 3, 6, 9, 12 bulan).
  + Area teks "Estimasi Cicilan per Bulan" yang ter-update secara otomatis.
  + Tombol "Lanjutkan".
* Behavior:
  + Interaktivitas: Setiap kali user mengubah nilai slider jumlah pinjaman atau memilih tenor yang berbeda, nilai "Estimasi Cicilan per Bulan" akan dihitung ulang secara *real-time* di sisi klien (misal: cicilan = (jumlah + (jumlah \* bunga \* tenor)) / tenor).
  + Validasi: Tombol "Lanjutkan" akan aktif hanya jika jumlah dan tenor sudah dipilih.
  + Navigasi: Setelah tombol "Lanjutkan" diklik, aplikasi akan membawa user ke layar Konfirmasi Pinjaman, dengan membawa data jumlah dan tenor yang dipilih.

**3. Screen: Konfirmasi Pinjaman (Loan Confirmation Screen)**

* UI Elements:
  + Ringkasan pengajuan: "Jumlah Pinjaman: Rp 10.000.000", "Tenor: 12 bulan", "Estimasi Cicilan: Rp 950.000/bulan".
  + Checkbox persetujuan "Saya telah membaca dan menyetujui Syarat & Ketentuan yang berlaku."
  + Tombol "Ajukan Pinjaman Sekarang".
* Behavior:
  + Tombol "Ajukan Pinjaman Sekarang" awalnya nonaktif. Tombol ini akan menjadi aktif setelah user mencentang checkbox persetujuan.
  + Saat tombol diklik, aplikasi akan menampilkan *loading spinner* dan memanggil API POST /api/v1/loans/apply.
  + Jika API merespons sukses, user akan diarahkan ke halaman "Pengajuan Terkirim".
  + Jika API merespons error (misal, karena koneksi terputus atau error server), akan ditampilkan pesan error yang informatif (contoh: "Gagal mengirim pengajuan. Silakan coba lagi.").