**Dokumen Rancangan Proyek TUKANGIN**

**1. Deskripsi Proyek**

**TUKANGIN** adalah platform marketplace layanan onsite yang menghubungkan pelanggan dengan penyedia jasa (tukang bangunan, tukang listrik, tukang servis, dsb). Proyek ini bertujuan menyediakan ekosistem terpercaya dengan:

* **Transaksi Aman** melalui sistem escrow.
* **Verifikasi Penyedia** untuk menjamin kualitas layanan.
* **Fleksibilitas Pemesanan** melalui dua mode layanan: fixed-price (langsung checkout) dan konsultatif (penawaran setelah konsultasi).
* **Dashboard Manajemen** untuk setiap peran: admin, provider, dan customer.

**2. Tujuan dan Ruang Lingkup**

**Tujuan Utama:**

* Menyediakan interface yang intuitif bagi pelanggan untuk mencari, memesan, dan melakukan konsultasi dengan penyedia jasa.
* Membantu penyedia mengelola layanan, jadwal, dan transaksi melalui dashboard mereka.
* Memberikan admin alat monitoring dan kontrol untuk mengawasi aktivitas platform, termasuk audit log dan penanganan sengketa.

**Ruang Lingkup:**

* Pengembangan arsitektur monolitik modular dengan struktur terpisah per fitur di dalam satu monorepo.
* Pembuatan API RESTful sebagai jembatan antara frontend dan backend.
* Implementasi server-side rendering (SSR) pada Next.js menggunakan App Router.
* Penerapan middleware keamanan tambahan (rate limiter, helmet, CORS, validasi input, dll).
* Integrasi autentikasi dengan JWT dan proteksi melalui 2FA untuk akun sensitif.
* Modul audit log yang dapat diakses oleh admin untuk melacak aktivitas sistem.
* Komponen teknis tambahan seperti notifikasi, sistem escrow, manajemen dispute, chat internal, dan sistem ulasan.

**3. Arsitektur Sistem**

**3.1. Arsitektur Monolitik Modular**

* **Monolitik Modular:** Seluruh aplikasi (frontend dan backend) berjalan dalam satu repositori dengan struktur modul terpisah per fitur. Meskipun dibangun sebagai satu kesatuan, modul-modul (seperti auth, user, provider, order, payment, chat, dsb) disusun sedemikian rupa agar mudah di-maintain, diuji, dan di-deploy.
* **Keunggulan:** Sederhana untuk tahap awal pengembangan, pengelolaan dependensi lebih mudah, serta integrasi antar modul yang lebih erat.
* **Skalabilitas:** Seiring pertumbuhan aplikasi, modul-modul yang paling berat (misal payment, chat) dapat dipisahkan secara bertahap menjadi microservices jika diperlukan.

**3.2. Setup Monorepo**

* **Monorepo Structure:** Satu repositori yang mengelola dua aplikasi utama:
  + **Frontend:** Aplikasi Next.js dengan App Router.
  + **Backend:** Aplikasi Express.js yang menyediakan API RESTful.
* **Contoh Struktur Folder:**

graphql

CopyEdit

/ (Root Repository)

├─ frontend/ # Aplikasi Next.js

│ ├─ app/ # App Router (Next.js 13+)

│ ├─ components/ # UI Components terpisah per modul fitur

│ ├─ lib/ # Helper dan API client (SWR, fetcher)

│ ├─ public/ # Assets statis

│ ├─ styles/ # Style CSS/SCSS/Tailwind

│ └─ next.config.js # Konfigurasi Next.js

│

└─ backend/ # Aplikasi Express.js

├─ src/

│ ├─ api/ # Modul API per fitur

│ │ ├─ auth/ # Endpoint autentikasi (login, register, 2FA, dll)

│ │ ├─ user/ # Manajemen user & role (admin, provider, customer)

│ │ ├─ provider/ # Endpoint pengelolaan layanan provider

│ │ ├─ order/ # Endpoint order dan transaksi (fixed & konsultatif)

│ │ ├─ payment/ # Integrasi pembayaran & escrow

│ │ ├─ chat/ # Fitur komunikasi internal (chat)

│ │ ├─ review/ # Ulasan dan rating

│ │ └─ admin/ # Endpoint khusus admin (audit log, dispute, laporan)

│ ├─ config/ # Konfigurasi (database, environment, logger)

│ ├─ middleware/ # Middleware (JWT auth, rate limiter, helmet, CORS, validasi)

│ ├─ models/ # Definisi skema database (ORM/ODM)

│ ├─ services/ # Business logic (notifikasi, pembayaran, audit log)

│ ├─ utils/ # Utility/helper functions

│ └─ app.js # Inisialisasi Express

├─ .env # Konfigurasi lingkungan (tidak di-commit)

└─ package.json # Dependencies dan script build/start

**4. API RESTful**

**4.1. Desain Endpoint Utama**

* **Autentikasi (Auth):**
  + POST /api/auth/register: Registrasi user baru
  + POST /api/auth/login: Login dengan pengiriman JWT
  + POST /api/auth/2fa: Verifikasi dua faktor untuk akun sensitif
  + POST /api/auth/refresh: Refresh token
* **User & Role Management:**
  + GET /api/users/:id: Dapatkan detail user
  + PUT /api/users/:id: Update profil
  + DELETE /api/users/:id: Hapus user (khusus admin)
* **Provider & Layanan:**
  + GET /api/providers: List penyedia
  + POST /api/providers: Registrasi/Update profil provider
  + GET /api/services: List layanan (fixed & konsultatif)
  + POST /api/services: Provider menambahkan layanan baru
  + PUT /api/services/:id: Update layanan
* **Order & Transaksi:**
  + POST /api/orders: Membuat order baru (penanganan fixed atau konsultatif)
  + GET /api/orders/:id: Detail order
  + PUT /api/orders/:id/confirm: Konfirmasi order selesai (trigger release escrow)
  + POST /api/orders/:id/dispute: Buka dispute jika ada sengketa
* **Pembayaran:**
  + POST /api/payment/initiate: Inisiasi pembayaran (integrasi gateway dan escrow)
  + POST /api/payment/webhook: Endpoint untuk menerima notifikasi pembayaran
* **Chat & Notifikasi:**
  + POST /api/chat/send: Mengirim pesan antar user
  + GET /api/chat/:orderId: Mengambil riwayat chat untuk order tertentu
  + GET /api/notifications: Fetch notifikasi user
* **Admin (Manajemen Sistem):**
  + GET /api/admin/audit-log: Ambil log audit seluruh aktivitas
  + PUT /api/admin/users/:id/role: Perubahan peran user
  + GET /api/admin/reports: Dasbor laporan statistik, transaksi, dispute

**4.2. Middleware API**

Setiap request ke API akan melalui middleware:

* **JWT Authentication Middleware:** Memvalidasi token, lalu menambahkan data user (id, peran) ke request.
* **Role-based Authorization Middleware (RBAC):** Memeriksa akses berdasarkan peran (admin, provider, customer).
* **Input Validation Middleware:** Menggunakan library seperti Joi/Zod/express-validator untuk memastikan data request valid dan aman.
* **Rate Limiting:** Menghindari abuse API dengan batasan request per IP.
* **Helmet:** Mengatur header keamanan HTTP.
* **CORS:** Konfigurasi agar hanya origin frontend yang diizinkan.
* **Error Handling Middleware:** Menangani error secara terpusat dan mengirim respons yang konsisten.
* **Audit Logging Middleware (khusus admin):** Mencatat aktivitas penting (login, perubahan data, transaksi) ke modul audit log.

**5. Rendering Frontend dengan Next.js**

**5.1. Strategi Rendering**

* **SSR dengan App Router:**
  + Halaman dinamis (dashboard, profil, order) ditangani dengan server-side rendering melalui Next.js App Router (app/ directory).
  + Data diambil secara server-side menggunakan getServerSideProps atau pendekatan App Router data fetching (React Server Components) untuk memastikan data terkini saat page load.
* **SSG/ISR:**
  + Halaman statis (tentang, FAQ, kebijakan) dapat menggunakan SSG dan secara berkala di-*regenerate* dengan ISR.
* **Client-Side Fetching:**
  + Untuk data yang berubah sangat cepat (notifikasi, chat), gunakan SWR untuk revalidasi data di client-side.

**5.2. Komunikasi Frontend-Backend**

* **API Client:** Buat helper utilitas (misalnya lib/api.js) untuk mengkonsumsi REST API Express. Gunakan SWR untuk caching dan revalidasi otomatis.
* **Auth Handling:** Data autentikasi (JWT) dikelola dengan HttpOnly cookies. Gunakan React Context atau state management (misal, Redux atau Zustand) untuk menyimpan status user.
* **Routing dan Proteksi Halaman:** Rute Next.js dilindungi dengan middleware (bisa menggunakan NextAuth atau custom route guarding) berdasarkan status autentikasi dan peran pengguna.

**6. Komponen Manajemen Peran**

**6.1. Dashboard Admin**

* **Audit Log:** Menampilkan daftar aktivitas penting user, transaksi, dan tindakan admin.
* **Manajemen Users:** CRUD untuk user, atur peran (provider, customer, admin), suspend atau hapus akun.
* **Manajemen Konten:** Moderasi profil provider, layanan, ulasan.
* **Laporan & Analitik:** Statistik transaksi, performa layanan, diskusi dispute.

**6.2. Dashboard Provider**

* **Manajemen Layanan:** CRUD untuk layanan (baik fixed maupun konsultatif), penjadwalan ketersediaan.
* **Order Management:** List order masuk, status order, update progress pengerjaan.
* **Keuangan:** Riwayat transaksi, request penarikan dana.
* **Chat & Notifikasi:** Fitur komunikasi langsung dengan pelanggan.

**6.3. Dashboard Customer**

* **Pencarian & Pemesanan:** Fitur pencarian layanan dengan filter kategori, rating, lokasi.
* **Manajemen Order:** Riwayat pemesanan, status order, chat dengan provider.
* **Ulasan & Rating:** Memberikan feedback, rating, ulasan pada provider.

**7. Keamanan Tambahan**

**7.1. Autentikasi dan 2FA**

* **JWT untuk Autentikasi Stateless:** Token dikeluarkan saat login dan disimpan di cookie HttpOnly.
* **2FA (Two-Factor Authentication):** Implementasi opsi 2FA untuk login akun sensitif, terutama bagi admin dan provider. Penggunaan OTP (SMS/email) atau aplikasi authenticator, yang harus diverifikasi sebelum mengizinkan akses ke fitur kritis.

**7.2. Audit Log**

* **Audit Logging di Admin:** Setiap aktivitas kritis (login, perubahan data profil, transaksi, dispute) dicatat melalui middleware audit log. Log disimpan di database khusus atau modul service dan dapat diakses oleh admin pada dashboard audit log.
* **Format Log:** Catatan harus mencakup userID, peran, aktivitas, timestamp, dan hasil aksi.

**7.3. Validasi Input dan Sanitasi**

* **Input Validation:** Gunakan library validasi (misalnya, Joi atau Zod) pada setiap endpoint API.
* **Sanitasi Input dan Output:** Gunakan mekanisme sanitasi untuk mencegah injection dan XSS (misalnya, express-mongo-sanitize, helmet).

**8. Komponen Teknis Tambahan**

* **Notifikasi:** Modul notifikasi real-time melalui email, SMS, dan push notification (misalnya, Firebase Cloud Messaging) untuk update status order, penawaran, dan reminder jadwal.
* **Sistem Escrow:** Implementasi modul escrow di mana pembayaran pelanggan ditahan sampai pekerjaan diselesaikan dan diverifikasi. Fungsi ini terintegrasi dalam alur order pada API pembayaran.
* **Manajemen Dispute:** Modul untuk mengelola sengketa antara pelanggan dan provider, dengan kemampuan admin untuk memantau, meninjau, dan memutuskan.
* **Chat Internal:** Real-time chat engine untuk komunikasi antara pelanggan dan provider. Chat disimpan (dengan enkripsi) untuk referensi sebagai bukti jika terjadi sengketa.
* **Sistem Ulasan & Rating:** Modul untuk mengelola ulasan, rating, dan sistem reputasi, dengan opsi moderasi oleh admin.
* **Manajemen Komisi & Fee:** Modul penghitungan otomatis fee platform pada setiap transaksi yang berhasil, serta laporan keuangan terintegrasi.

**9. Strategi Deployment dan Skalabilitas**

* **Containerization:** Gunakan Docker untuk mengemas aplikasi frontend dan backend secara terpisah dalam container.
* **CI/CD Pipeline:** Otomatiskan build, testing, dan deployment menggunakan tool seperti GitHub Actions atau GitLab CI.
* **Monitoring:** Implementasikan monitoring (misalnya, Prometheus, Grafana) untuk log, performa, dan penggunaan resource.
* **Auto-Scaling:** Konfigurasi load balancer dan pengaturan scaling secara horizontal (multi-instance) untuk menangani trafik tinggi.
* **Environment Management:** Pisahkan file konfigurasi environment untuk development, staging, dan production; gunakan secret manager untuk menyimpan API keys dan credential penting.
* **Zero Downtime Deployment:** Gunakan strategi blue-green atau canary deployment untuk mencegah gangguan saat melakukan pembaruan.

**Penutup**

Dokumen rancangan ini menjadi pedoman komprehensif untuk membangun TUKANGIN dari awal secara bertahap dengan pendekatan Next.js (frontend) dan Express.js (backend) yang terintegrasi dalam struktur monorepo modular. Setiap komponen mulai dari autentikasi, pengelolaan peran, API RESTful, rendering server-side, hingga middleware dan komponen teknis tambahan, dirancang untuk membangun platform yang aman, scalable, dan mudah dikelola. Pedoman ini diharapkan memudahkan tim developer dalam mengimplementasikan setiap fitur secara konsisten dan berkualitas.

Silakan digunakan sebagai acuan utama untuk langkah-langkah pengembangan berikutnya. Jika ada pertanyaan atau bagian yang perlu didiskusikan lebih lanjut, silakan beri masukan untuk penyesuaian lebih lanjut.