Arsitektur Frontend Sistem Pembelajaran SAT/UTBK Berbasis Agentic AI

1. Prinsip Desain dan Filosofi Arsitektur (Ref. Fase 1 & 2)

A. Prinsip UI/UX Inti

1. **Responsif & Mobile-First:** Wajib responsif. Prioritas pada UI yang bersih dan fungsional di perangkat mobile, terutama untuk sesi latihan soal dan interaksi Chat AI.
2. **Aksesibel (A11y):** Memastikan kepatuhan WCAG untuk pengalaman yang inklusif.
3. **Performa Tinggi & Interaktif:** Memanfaatkan **Next.js (SSR/SSG)** untuk pemuatan cepat. Mengutamakan pemisahan *chunk* code (lazy loading) dan optimasi aset agar UI responsif terhadap *input* pengguna dan *feedback* dari Agentic AI.
4. **Visualisasi Data Canggih:** Desain harus mampu menampilkan hasil tes dan metrik pembelajaran adaptif (misalnya, grafik kemajuan per subtopik, rekomendasi AI) secara intuitif dan menarik.

B. Filosofi Arsitektur: Modular dan Berorientasi Fitur (Feature-Sliced Design - FSD)

Struktur modular memudahkan isolasi fitur, memungkinkan tim untuk dengan cepat mengintegrasikan perubahan API dari Microservices Backend (terutama **Agent Orchestrator Service** dan **Assessment Service**)

2. Pilihan Teknologi (Ref. Fase 5)

Area	Pilihan Teknologi	Alasan (Sinkronisasi Backend & Tema AI)
Framework Frontend	**Next.js (React Framework)**	Ideal untuk *SEO* (Landing pages), *Routing* terstruktur, dan performa tinggi melalui SSR/SSG.
Sistem Styling	**Tailwind CSS**	Pengembangan UI yang sangat cepat dan konsisten, penting untuk iterasi cepat pada visualisasi AI.
State Management	**Zustand + React Query**	**Zustand** untuk state UI global. **React Query** adalah **KRITIS** untuk manajemen *Server State* (caching,

		sinkronisasi) dari **Data Service** dan **Question Service** Backend.
Visualisasi Data	**Recharts/D3.js (dalam komponen)**	Untuk rendering grafik dan visualisasi kompleks *Adaptive Scoring* dan metrik dari **Analytics Service**.
API Client	**Axios**	Komunikasi terpusat dengan **API Gateway**, mengelola JWT/Refresh Token dari **Auth Service** dan interceptor untuk rate limiting.

2. Pilihan Teknologi (Ref. Fase 5)

Area	**Pilihan Teknologi**	**Alasan (Sinkronisasi Backend & Tema AI)**
Framework Frontend	**Next.js (React Framework)**	Ideal untuk *SEO* (Landing pages), *Routing* terstruktur, dan performa tinggi melalui SSR/SSG.
Sistem Styling	**Tailwind CSS**	Pengembangan UI yang sangat cepat dan konsisten, penting untuk iterasi cepat pada visualisasi AI.
State Management	**Zustand + React Query**	**Zustand** untuk state UI global. **React Query** adalah **KRITIS** untuk manajemen *Server State* (caching, sinkronisasi) dari **Data Service** dan **Question Service** Backend.
Visualisasi Data	**Recharts/D3.js (dalam komponen)**	Untuk rendering grafik dan visualisasi kompleks *Adaptive Scoring* dan metrik dari **Analytics Service**.
API Client	**Axios**	Komunikasi terpusat dengan **API Gateway**, mengelola JWT/Refresh Token dari **Auth Service** dan interceptor untuk rate limiting.

3. Struktur Folder Modular (Feature-Based) (Ref. Fase 4)

Struktur ini memetakan Feature Frontend ke Microservices Backend yang relevan, terutama untuk modul AI.

src/

```
├── app/                # Next.js App Router (Routing, Global Layouts)
├── assets/             # Aset statis (Gambar, Font, Ikon SVG)
├── components/         # Komponen UI yang dapat digunakan kembali (Reusable)
│   ├── base/           # Button, Input, Card, Modal
│   ├── layout/         # Sidebar, Header, Footer
│   └── complex/        # DataTable, **AdaptiveChart**, LoadingSpinner
├── features/           # Modul aplikasi berorientasi fitur (Domain Bisnis)
│   ├── auth/           # Login/Logout (Auth Service)
│   ├── dashboard/      # Ringkasan Kinerja (Analytics Service)
│   ├── practice/       # Latihan Soal, UI Soal Adaptif (Question Service)
│   ├── results/        # Tampilan Hasil Tes & Analisis **Adaptive Scoring** (Assessment Service)
│   ├── ai-chat/        # UI Chat, Integrasi Langsung **Agent Orchestrator Service**
│   └── admin-panel/    # Manajemen Konten & Laporan (CMS/Admin Panel Service)
├── hooks/              # Custom React Hooks (useAuth, useFetch, useWebSocket)
├── services/           # Layer komunikasi data (API Client & Real-time)
│   ├── api/            # Konfigurasi Axios dan functions CRUD per Microservice
│   ├── realtime/       # Logika koneksi WebSocket (Notification & Progress)
│   ├── data-models/    # Definisi TypeScript Interfaces/Types (Sangat penting untuk AI payload)
│   └── utils/          # Helper functions
└── store/              # State Management (Zustand Stores)
    ├── authStore.ts    # Global state otentikasi & info user
    ├── notificationStore.ts # Menangani notifikasi real-time dari Backend.
    ├── **aiFeedbackStore.ts** # Menyimpan state sesi AI Chat dan riwayat interaksi.
```

└── uiStore.ts # Pengaturan UI global.

4. Pengelolaan Status dan Interaksi AI (Ref. Fase 4)

A. Penggunaan React Query untuk Server State

Kami menggunakan React Query untuk mengelola data yang sering diperbarui (Soal adaptif) dan hasil tes. Ini mengurangi *boilerplate* fetching dan memastikan data UI selalu sinkron dengan Microservices Backend.

B. aiFeedbackStore.ts

Store khusus ini menangani interaksi dengan **Agent Orchestrator Service**:

1. Menyimpan *chat history* dan konteks percakapan untuk AI Tutor.
2. Menyimpan status isTyping atau isGeneratingResponse untuk memberikan feedback visual yang lancar kepada pengguna (mengingat latensi pemrosesan AI).
3. Mengelola **streaming response** (jika didukung API Gateway), di mana teks dari AI muncul secara bertahap, bukan menunggu respons penuh.

C. Alur Data untuk Soal Adaptif

1. Siswa menyelesaikan soal. Komponen memanggil Aksi.
2. **Service Layer** memanggil Endpoint submitAnswer (Assessment Service).
3. Backend memproses jawaban, memperbarui *Adaptive Scoring*, dan memicu **Question Service** untuk menghasilkan soal berikutnya berdasarkan Vector DB.
4. Frontend menggunakan **React Query** untuk *invalidate* query soal lama dan secara otomatis me-fetch soal baru (useQuery('currentQuestion')), yang memastikan soal adaptif dimuat dengan cepat.

5. Komunikasi API dan Keamanan (Ref. Fase 4 & 8)

A. API Client Centralization & Handling Agentic AI

Semua komunikasi diarahkan melalui **API Gateway**.

- **API Chat:** Endpoint ke **Agent Orchestrator Service** memerlukan *timeout* yang lebih panjang atau dukungan streaming, karena pemrosesan agent AI dapat memakan waktu lebih lama daripada permintaan REST standar.
- **Keamanan:** Interceptor Axios wajib mengelola **JWT dengan Refresh Token** (sesuai Auth Service) untuk menjaga sesi pengguna tetap aman dan terautentikasi.

B. Pemodelan Data (TypeScript)

Definisi tipe data yang ketat sangat penting, terutama untuk payload kompleks seperti:

- IQuestion: Termasuk *embedding* ID atau metadata vektor.
- IAdaptiveReport: Struktur data untuk visualisasi (misalnya, skill matrix, confidence score).

6. Strategi Implementasi dan Migrasi (Ref. Fase 6 & 7)

Fokus implementasi harus selalu pada **Core Learning Loop** dan **AI Integration**.

Fase Migrasi (Sprint)	**Fitur Utama (MVP)**	**Keterkaitan Backend**
Sprint 1 (Otentikasi & Dasar)	Login/Register, Global Layout, Dashboard Dasar.	Auth Service, User Service.
Sprint 2 (Core Adaptive)	**Latihan Soal** (UI Soal, Timer, Pengumpulan Jawaban), Navigasi.	Question Service, Assessment Service.
Sprint 3 (AI Tutor)	**Chat AI** (UI chat, koneksi ke Agent Orchestrator), Notifikasi Real-Time.	Agent Orchestrator Service, Notification Service (via WebSockets).
Sprint 4 (Advanced Analytics)	**Hasil Tes Lengkap** dengan **Visualisasi Adaptif** (Recharts/D3.js).	Analytics Service, Vector DB Service.
Sprint 5+ (Enhancement)	Leaderboard, Gamifikasi, Admin Panel Penuh, Optimasi Performa.	CMS/Admin Panel Service, Fitur Tambahan.