Tugas Perancangan API

II3160 – Teknologi Sistem Terintegrasi

Disusun Oleh:

Brandon Theodore Freinov (18223020)
Derick Amadeus Budiono (18223090)

Sekolah Teknik Elektro dan Informatika
Program Studi Sistem dan Teknologi Informasi
Semester IV - 2025/2026
Institut Teknologi Bandung
2025

Daftar Isi

Daftar Gambar

1. Deskripsi Sistem

Lanskap hiburan digital saat ini sangat terfragmentasi di berbagai platform *streaming* dan basis data yang terpisah-pisah, menyulitkan pengguna melacak tontonan mereka. Pengguna sering kali kesulitan mengingat detail episode terakhir yang ditonton karena data tontonan terisolasi di dalam aplikasi spesifik seperti Netflix atau layanan lokal. Ketiadaan interoperabilitas ini memaksa pengguna untuk mengingat progres mereka secara manual atau menggunakan catatan yang tidak terstruktur. Akibatnya, timbul kebutuhan signifikan akan sebuah platform terpusat yang mampu mengagregasi metadata dari berbagai domain hiburan ke dalam satu antarmuka tunggal. Sistem ini dirancang sebagai jembatan universal yang menghubungkan pengguna dengan beragam jenis media animasi melalui mekanisme pelacakan yang terstandarisasi.

Gambar 1.1. *Good Job*

Sebagai solusi, kami mengusulkan "AniLog", sebuah sistem terintegrasi yang berfungsi sebagai agregator komprehensif untuk film, serial TV, dan anime. Inti dari sistem ini bergantung pada arsitektur *RESTful API* yang kuat, yang bertindak sebagai tulang punggung pertukaran data antara *client* dan *server*. API ini memungkinkan pengembang untuk mengakses basis data terkurasi guna membangun aplikasi *frontend* yang beragam atau alat analisis khusus. Skema basis data dirancang secara

relasional untuk menangani hubungan kompleks antara judul, genre, dan staf produksi dengan performa tinggi. Selain itu, sistem menerapkan protokol validasi data yang ketat untuk memastikan integritas konten yang dikontribusikan oleh komunitas tetap terjaga.

Fungsionalitas utama sistem berfokus pada fitur pelacakan dan penyortiran yang *user-centric*, serupa dengan konsep MyAnimeList namun dengan cakupan integrasi lebih luas. Pengguna memiliki kemampuan untuk membuat daftar kustom, mengategorikan konten berdasarkan status seperti "Sedang Menonton", "Selesai", atau "Rencana Menonton". Sistem ini dilengkapi dengan mesin penyortiran canggih yang memproses kueri berdasarkan parameter jamak seperti popularitas, tanggal rilis, dan rata-rata rating pengguna. Untuk meningkatkan pengalaman pengguna, platform mendukung penandaan (*tagging*) granular yang memungkinkan hasil pencarian presisi sesuai dengan minat spesifik pengguna. Fitur-fitur ini secara kolektif mendorong terbentuknya ekosistem data yang rapi dan terstruktur bagi para penggemar media.

Aspek krusial dari AniLog adalah kemampuannya untuk berintegrasi dengan penyedia layanan eksternal guna memperkaya dataset internalnya secara otomatis. Sistem menggunakan *background workers* untuk secara berkala mengambil dan menyinkronkan metadata dari sumber eksternal terpercaya seperti API publik pihak ketiga. Integrasi ini memastikan bahwa platform selalu menampilkan informasi terkini mengenai jadwal rilis dan detail pemeran tanpa intervensi manual berlebihan. Di luar sekadar pengambilan data, sistem menyediakan *endpoint* yang aman bagi pengguna untuk menautkan akun eksternal mereka. Proses ini mengeliminasi redundansi entri data manual dan menjamin daftar pengguna tetap akurat di berbagai titik akses.

Komponen terakhir dari sistem terintegrasi ini adalah mesin rekomendasi sederhana yang memanfaatkan data interaksi pengguna untuk memberikan saran konten. Dengan menganalisis pola penilaian dan status tontonan, sistem dapat menyoroti judul-judul relevan yang mungkin terlewatkan oleh pengguna. Modul ini berjalan sebagai layanan pendukung yang memproses data statistik rating untuk menghasilkan papan peringkat global yang dinamis. Algoritma tersebut membandingkan agregat skor dari seluruh basis data untuk mengidentifikasi tren popularitas secara real-time.

Pendekatan berbasis data ini mentransformasi platform dari sekadar alat pelacak pasif menjadi pusat informasi yang adaptif.

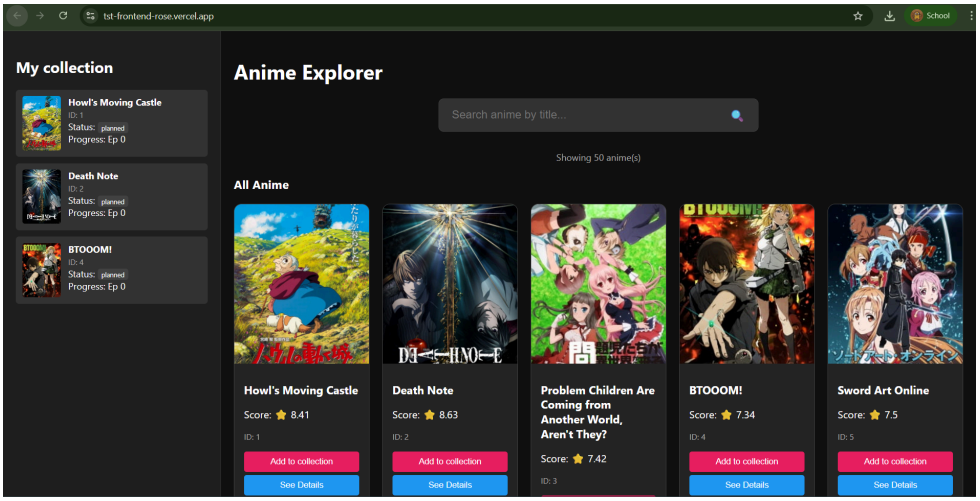
2. Arsitektur Layanan

2.1. Deskripsi Arsitektur Integrasi

Sistem ini dibangun menggunakan *static web* berbasis html yang berjalan di sisi client (*client side*). Arsitektur yang digunakan adalah **Microservices Aggregation Pattern**, di mana aplikasi web bertindak sebagai orkestrator yang memanggil beberapa *endpoint* API dari server yang berbeda secara asinkron (AJAX/Fetch).

Dalam skenario penggunaan, ketika pengguna membuka halaman pencarian anime:

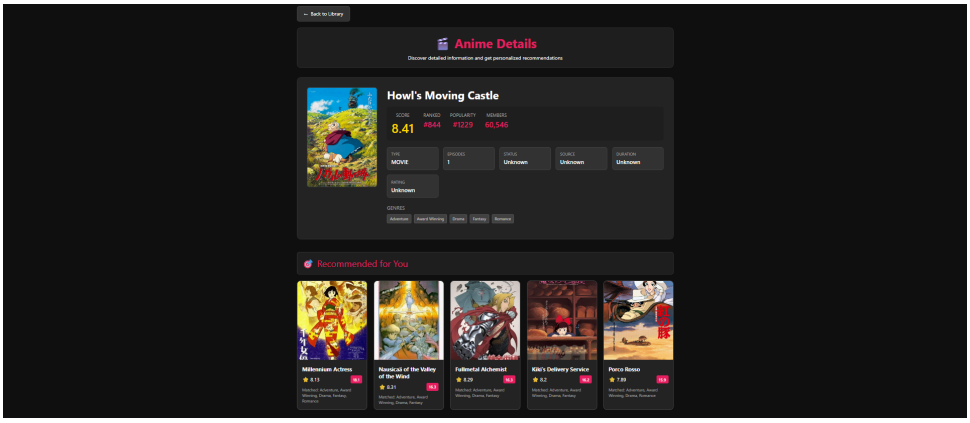
1. AniLog akan mengirim permintaan HTTP GET ke Service 1 (Port 6969) untuk mengambil Metadata Anime (Judul, Genre, Skor).



Gambar 2.1. *Anime List Page*

2. Secara paralel, AniLog mengirim permintaan HTTP GET ke Service 2 (Port [6767]) untuk mengambil data pengguna dan log anime yang tersimpan.

3. Ketika user ingin melihat lebih detail anime, AniLog akan mengirim permintaan HTTP GET ke Service 1 (Port 6969) untuk mengembalikan ringkasan anime secara detail & *recommendation* untuk anime serupa.



Gambar 2.2. *Anime Detail Page*

Data dari kedua sumber digabungkan (merged) di browser pengguna dan ditampilkan menjadi satu tampilan halaman yang utuh.

2.2. Alur Data

1. Inisialisasi Halaman (Page Load):
    a. Ketika pengguna membuka halaman utama, sistem secara paralel memanggil dua fungsi: performSearch() dan loadcollection().
    b. Browser mengirim permintaan ke Service 1 (AnimeList) untuk mendapatkan daftar anime default ("All Anime") guna ditampilkan di bagian utama (main grid).
    c. Browser mengirim permintaan ke Service 2 (Tracker) untuk mengambil daftar watchlist pengguna berdasarkan userId.
2. Agregasi Data Koleksi (Sidebar):
    a. Service 2 mengembalikan daftar JSON yang berisi itemId (ID Anime) dan status tontonan.
    b. Service 1 mengembalikan detail lengkap (Judul, Gambar, Skor) untuk kemudian di *render* pada sidebar "My Collection". Inilah proses integrasi data yang terjadi di sisi klien.
3. Proses Pencarian (Search):
    a. Saat pengguna mengetik di kolom pencarian, fungsi debounce diaktifkan.
    b. Aplikasi memanggil Service 1 dengan parameter query (?q=judul).
    c. Hasil pencarian diperbarui secara real-time di layar utama

4. Aksi Penambahan (Add to Collection):
    a. Pengguna mengklik tombol "Add to collection" pada kartu anime.
    b. Aplikasi mengirim permintaan POST ke Service 2 yang berisi userId, animeId, dan status awal "watching".
    c. Setelah berhasil, aplikasi memuat ulang sidebar (mengulangi langkah nomor 2) untuk memperbarui tampilan daftar koleksi.

3. Penggunaan API

Sistem integrasi ini memanfaatkan sejumlah *endpoint* dari kedua layanan mikro. Berikut adalah spesifikasi penggunaan API dalam aplikasi:

3.1. Service 1: AniLog (Penyedia Data Anime)

Layanan ini bertindak sebagai sumber kebenaran (*source of truth*) untuk metadata konten.

- **Base URL:** http://udinbanda.theokaitou.my.id
- **Endpoints:**
    1. GET /anime: Mengambil seluruh daftar anime (digunakan untuk tampilan awal).
    2. GET /anime?q={query}: Melakukan pencarian anime berdasarkan judul.
    3. GET /anime/{id}: Mengambil detail spesifik satu anime (Judul, Gambar, Skor) berdasarkan ID.

3.2. Service 2: Tracker (Manajemen User & Koleksi)

Layanan ini bertindak sebagai penyimpan *state* atau aktivitas pengguna.

1. **Base URL:** http://saber.theokaitou.my.id
2. **Endpoints:**

3.3. GET /collection/{userId}: Mengambil daftar koleksi anime milik pengguna tertentu. Respons berupa *array* objek yang berisi itemId, status, dan progress.

3.4. POST /collection/add: Menambahkan anime baru ke dalam koleksi pengguna. Membutuhkan *body* JSON berupa { userId, animeId, status }.

4. Deployment

4.1. Metode Deployment

Arsitektur sistem memisahkan antara *backend* (API) dan *frontend* (UI), strategi *deployment* dilakukan secara terpisah untuk menjaga performa dan skalabilitas:

1. **Backend Microservices:** Kedua layanan *backend* (AniLog dan Tracker) dijalankan menggunakan kontainerisasi (Docker) pada server/STB masing-masing. Akses publik dimungkinkan melalui mekanisme *tunneling* (Cloudflare Tunnel) yang memetakan *port* lokal (6969 dan 6767) ke domain publik (`udinbanda...` dan `saber...`). Hal ini memungkinkan API diakses dari jaringan internet manapun tanpa konfigurasi IP statis yang rumit.
2. **Frontend Integrasi:** Aplikasi antarmuka pengguna dibangun sebagai *Static Web Application* (hanya terdiri dari HTML, CSS, dan JS). Oleh karena itu, *deployment* dilakukan menggunakan layanan *hosting* statis **Vercel**. Metode ini dipilih karena ringan, cepat, dan tidak membebani sumber daya server *backend* untuk me-*render* tampilan. Klien (Browser) hanya perlu mengunduh file statis sekali, dan selanjutnya komunikasi data dilakukan langsung antara Browser dan API Backend.

4.2. URL Layanan

Berikut adalah tautan akses untuk layanan yang telah diimplementasikan:
URL Aplikasi Integrasi (Frontend):
https://tst-frontend-rose.vercel.app/

Source Code Repository:
Frontend : https://github.com/DerickAmadeus/TST_Frontend
Microservice : https://github.com/DerickAmadeus/TST