Instruksi Tes Teknis Web Developer (Full Stack) — Single Page CRUD Akademik

Batas pengumpulan: 18 Februari 2026, 23:59 WIB (Asia/Jakarta) Link submission:
https://forms.gle/5C2CR8Nj16Zw1kEa9

1. Tujuan Tes

Menguji kemampuan kandidat dalam membangun sistem dengan **modern stack** (frontend + backend atau full stack) yang:

- Memiliki fitur **CRUD** lengkap.
- Melakukan **insert/update terkoordinasi ke 3 tabel yang saling berelasi** (FK) **dalam satu transaksi**.
- Mampu menangani **dataset besar** (minimal **5.000.000 baris**) dengan **server-side pagination, sorting, filtering, searching, dan export**.

2. Ruang Lingkup Bisnis (Proses Akademik)

Tema bisnis: **Proses KRS / Pengambilan Mata Kuliah**.

2.1 Entitas dan Relasi (Wajib 3 Tabel)

Gunakan minimal 3 tabel berikut (boleh menambah tabel lain bila perlu):

1) **students** (mahasiswa) 2) **courses** (mata kuliah) 3) **enrollments** (KRS/pengambilan MK)

Relasi yang wajib:

- `enrollments.student_id` -> `students.id`
- `enrollments.course_id` -> `courses.id`

2.2 Operasi CRUD (Wajib)

Aplikasi harus menyediakan halaman tunggal (single page) untuk mengelola data **KRS** (enrollments) dengan ketentuan:

- **Create**: saat membuat KRS, form **wajib melakukan insert ke 3 tabel** (students + courses + enrollments) **atau** melakukan upsert (contoh: student sudah ada, course sudah ada) namun tetap memastikan operasi melibatkan 3 entitas dan berjalan **atomic** (1 transaksi).
- **Read**: menampilkan data KRS dalam tabel dengan ketentuan di bagian 4.
- **Update**: dapat mengubah data enrollment, dan bila kandidat memilih, dapat mengubah data student/course terkait (jelaskan di README).

- **Delete**: menghapus enrollment (hard delete atau soft delete, jelaskan pilihan dan dampaknya).

> ***Catatan: kandidat bebas menentukan apakah form Create meminta input student & course sebagai "data baru" atau "pilih yang sudah ada + opsi tambah baru", yang penting: saat Create, tiga tabel terlibat dan FK valid.***

3. Teknologi (Bebas, tetapi Modern)

Kandidat bebas memilih stack, namun wajib memenuhi:

- Frontend modern: React/Vue/Svelte/Angular (boleh Next/Nuxt/Vite, dsb).
- Backend modern: Node (Express/Fastify/Nest), Laravel, Django/FastAPI, Spring Boot, .NET, atau sejenis.
- DB relasional: PostgreSQL/MySQL/MariaDB (disarankan PostgreSQL).
- Wajib ada dokumentasi cara menjalankan (local) dan akses online.

4. Kebutuhan UI Tabel Data (Wajib)

Data **enrollments** harus ditampilkan dalam bentuk tabel dengan kolom minimal berikut (boleh lebih):

- NIM (student_nim)
- Nama Mahasiswa (student_name)
- Kode MK (course_code)
- Nama MK (course_name)
- Semester (semester)
- Tahun Ajaran (academic_year)
- Status (status)

4.1 Server-Side Pagination (Wajib)

- Pagination harus **server-side**: query ke backend menerima page dan `page_size` (atau offset/limit).
- Tabel harus tetap responsif pada data 5 juta baris.

4.2 Sorting per Header (Wajib)

- **Setiap header kolom** wajib bisa di-sort (ASC/DESC) dengan eksekusi di backend.
- UI harus menampilkan indikator sort.

4.3 Quick Filter (Minimal 2, Wajib)

Minimal 2 quick filter, contoh:

- Filter **Status**: Draft / Submitted / Approved / Rejected

- Filter **Semester**: Ganjil / Genap (atau 1/2)

Quick filter harus dieksekusi server-side.

4.4 Live Searching (Minimal 3 Kolom Utama, Wajib)

Wajib ada input pencarian yang melakukan pencarian **real-time** (debounce 300–500ms) minimal pada 3 kolom utama, contoh:

- NIM
- Nama Mahasiswa
- Kode MK

Pencarian harus dieksekusi server-side.

4.5 Advanced Filter untuk Semua Kolom (Wajib)

Aplikasi wajib memiliki advanced filter yang bisa mengatur filter **untuk setiap kolom yang ditampilkan**, dan dapat digunakan **secara bersamaan** (multi filter).

Contoh kemampuan:

- `student_nim` “contains/startsWith/equal”
- `academic_year` “equal/between”
- `semester` “in”
- `status` “in”
- `course_code` “contains”

4.6 Advanced Order Multi Kolom (Wajib)

Aplikasi wajib mendukung advanced order:

- Multi kolom berurutan (mis. `academic_year DESC`, lalu `semester ASC`, lalu `student_nim ASC`)
- Mendukung logika **AND/OR** (jelaskan implementasi yang dipilih di README).

Contoh interpretasi:

- AND: kombinasi filter A *dan* B sekaligus
- OR: baris cocok filter A *atau* B (minimal untuk 2 filter)

> ***Catatan: Implementasi AND/OR biasanya terkait filter (bukan sorting). Kandidat boleh mengartikan “advanced order AND/OR” sebagai *kombinasi kondisi query* (filter group) dan menyertakan dokumentasi.***

4.7 Export Seluruh Data (5 Juta Baris) ke CSV/Excel (Wajib)

Wajib ada fitur export yang menghasilkan file **untuk seluruh data sesuai filter/query** (bukan hanya data halaman saat ini), termasuk skenario **5.000.000 baris**.

Ketentuan:

- Format minimal **CSV**. Excel/XLSX opsional (ingat limit baris XLSX; jika XLSX dipilih, boleh split beberapa sheet/zip).
- Export harus berjalan untuk 5 juta data (boleh streaming response atau job + download link, tetapi user tetap bisa mendapatkan file lengkap).
- Cantumkan strategi performa (index, streaming, chunking, queue) di README.

5. Validasi Form (Ketat di Frontend dan Backend) — Wajib

Validasi wajib ada di **frontend dan backend** (tidak boleh hanya salah satu).

5.1 Contoh aturan validasi minimal (boleh diperketat)

students

- `nim`: wajib, unik, 8–12 digit angka, tidak boleh spasi
- `name`: wajib, 3–100 karakter
- `email`: wajib, format email valid, unik

courses

- `code`: wajib, unik, format `[A-Z]{2,4}[0-9]{3}` (contoh: IF101)
- `name`: wajib, 3–120 karakter
- `credits`: wajib, integer 1–6

enrollments

- `academic_year`: wajib, format YYYY/YYYY (contoh 2025/2026)
- `semester`: wajib, enum (GANJIL/GENAP atau 1/2)
- `status`: wajib, enum (DRAFT/SUBMITTED/APPROVED/REJECTED)
- Unique constraint (disarankan): `(student_id, course_id, academic_year, semester)` tidak boleh duplikat

5.2 Error Handling (Wajib)

- Pesan error harus jelas di UI dan dari API (HTTP status code tepat).
- Backend wajib melakukan sanitasi input dan menolak payload invalid.

6. Data Seeder Minimal 5.000.000 Baris (Wajib)

Aplikasi wajib menyediakan seeder untuk menghasilkan **minimal 5.000.000 baris** data (disarankan untuk tabel enrollments).

Ketentuan:

- Seeder dapat dijalankan via command/script (contoh: `npm run seed`, `php artisan db:seed`, `python manage.py seed`, dsb).
- Sertakan parameterisasi (mis. jumlah baris) bila memungkinkan.
- Wajib menyertakan instruksi di README untuk:
- Membuat database
- Menjalankan migration
- Menjalankan seeder 5 juta data
- Membuktikan jumlah data (query count)

> ***Disarankan: gunakan bulk insert / COPY (PostgreSQL) / batching untuk mempercepat.***

---

7. Non-Functional Requirements (NFR)

- **Performa**: endpoint list (pagination + filter/sort) harus tetap responsif pada skala 5 juta data (gunakan index yang tepat).
- **Keamanan dasar**: validasi server-side, proteksi dari SQL injection (gunakan ORM/query parameterized), CORS aman.
- **Kualitas kode**: struktur rapi, linting/formatting, error logging.
- **Observability (opsional)**: basic request logging.

8. Deliverables (Wajib)

1) **Public Git Repository** berisi:

- Source code frontend & backend (atau monorepo).
- README lengkap (setup local, env vars, migrasi, seeding 5 juta, cara deploy).
- DB schema/migrations.
- (Opsional) Postman/Insomnia collection.

2) **URL Aplikasi yang sudah dideploy** dan dapat diakses online. 3) **URL/hasil export** dapat diuji (mis. endpoint export atau tombol export di UI).

9. Skenario Pengujian (Test Scenarios) — Wajib Dipenuhi

Gunakan daftar berikut sebagai acuan acceptance criteria. Kandidat **tidak perlu** menulis automated test, namun nilai plus bila ada.

TS-01: Setup & Seed 5 Juta Data

Langkah:

1. Clone repo.
2. Jalankan migrasi DB.
3. Jalankan seeder untuk menghasilkan 5.000.000 baris enrollments.
4. Jalankan query count.

Ekspektasi:

- Seeder selesai tanpa error.
- Jumlah data >= 5.000.000 (dibuktikan dengan `COUNT(*)`).
- Aplikasi tetap bisa berjalan dan menampilkan data.

TS-02: Create (Insert ke 3 Tabel dalam 1 Transaksi)

Langkah:

5. Buka form Create KRS.
6. Input data mahasiswa baru (nim, name, email).
7. Input data mata kuliah baru (code, name, credits).
8. Input data enrollment (academic_year, semester, status).
9. Submit.

Ekspektasi:

- Data tersimpan ke **students**, **courses**, **enrollments**.
- FK pada enrollments valid.
- Jika terjadi error pada salah satu insert, seluruh transaksi rollback (tidak ada data setengah masuk).
- UI menampilkan notifikasi sukses.

TS-03: Validasi Ketat (Frontend)

Langkah:

10. Isi NIM dengan huruf atau panjang < 8.
11. Isi course code tidak sesuai pola.
12. Kosongkan field wajib.
13. Submit.

Ekspektasi:

- Submit ditolak di frontend.
- Pesan error per field muncul dan jelas.

TS-04: Validasi Ketat (Backend)

Langkah:

14. Kirim request API Create dengan payload invalid (gunakan curl/postman).
15. Coba duplikasi nim atau course code.
16. Coba duplikasi enrollment unik (student, course, academic_year, semester).

Ekspektasi:

- API mengembalikan error 4xx dengan pesan jelas.
- Tidak ada data invalid masuk DB.

TS-05: Read - Tabel + Server-Side Pagination

Langkah:

17. Buka halaman tabel KRS.
18. Ubah page dan page size.
19. Pastikan request ke backend membawa parameter pagination.

Ekspektasi:

- Data berubah sesuai page.
- Backend hanya mengambil data sesuai page (bukan load semua).
- Total record ditampilkan (atau minimal total page).

TS-06: Sorting di Setiap Header Kolom (Server-Side)

Langkah:

20. Klik header kolom (mis. NIM) untuk sort ASC.
21. Klik lagi untuk sort DESC.
22. Coba kolom lain.

Ekspektasi:

- Setiap kolom bisa di-sort.
- Query sorting dilakukan di backend.
- UI menunjukkan arah sort.

TS-07: Quick Filter (Minimal 2)

Langkah:

23. Aktifkan quick filter Status.
24. Aktifkan quick filter Semester.
25. Kombinasikan keduanya.

Ekspektasi:

- Tabel hanya menampilkan data yang sesuai filter.
- Filter dieksekusi server-side.

TS-08: Live Searching (Minimal 3 Kolom)

Langkah:

26. Ketik NIM (parsial) di search box.
27. Ketik nama mahasiswa (parsial).
28. Ketik kode MK (parsial).

Ekspektasi:

- Hasil berubah real-time (dengan debounce).
- Pencarian dieksekusi server-side.
- Minimal 3 kolom utama tercakup.

TS-09: Advanced Filter Semua Kolom (Multi Filter)

Langkah:

29. Buka panel advanced filter.
30. Set filter untuk beberapa kolom sekaligus (mis. academic_year + status + course_code).
31. Terapkan.

Ekspektasi:

- Semua kondisi filter diterapkan.
- Query di backend benar (AND default).
- Dapat reset/clear filter.

TS-10: Advanced Query (AND/OR)

Langkah:

32. Buat filter group A dan B.
33. Terapkan kombinasi AND.
34. Terapkan kombinasi OR.

Ekspektasi:

- Hasil sesuai logika AND/OR yang didokumentasikan.
- Backend menangani parsing query dengan aman.

TS-11: Update

Langkah:

35. Pilih salah satu enrollment.
36. Ubah status atau semester/tahun.
37. Simpan.

Ekspektasi:

- Data berubah di DB.
- Validasi tetap berlaku.
- UI memperbarui baris data.

TS-12: Delete

Langkah:

38. Hapus salah satu enrollment.
39. Refresh data.

Ekspektasi:

- Enrollment tidak tampil lagi (atau soft delete, sesuai dokumentasi).
- Tidak merusak data student/course (kecuali kandidat memilih cascading dengan alasan jelas).

TS-13: Export 5 Juta Data (CSV/Excel)

Langkah:

40. Tanpa filter, klik Export.
41. Tunggu file siap/terunduh.
42. Verifikasi file berisi seluruh data (jumlah baris besar).

Ekspektasi:

- File mencakup seluruh dataset (atau seluruh dataset sesuai filter/query).
- Tidak terbatas pada 1 page.
- Mekanisme tetap stabil untuk 5 juta baris (stream/chunk/job).
- File dapat dibuka/diinspeksi (CSV disarankan; untuk XLSX jelaskan strategi split/zip).

10. Kriteria Penilaian (Rubrik)

Penilaian bersifat menyeluruh, dengan fokus:

- **Kebenaran fungsi** sesuai requirement (pagination/sort/filter/search/export).
- **Kualitas implementasi query server-side** (indexing, performa, keamanan).
- **Atomic transaction** insert ke 3 tabel.
- **Validasi ketat** FE + BE.
- **Kualitas UX** (responsif, error message jelas).

- **Kualitas kode & dokumentasi** (README, struktur, deploy, reproducible).
- **Kemampuan menangani 5 juta data** (seeding dan operasional UI/API).

11. Cara Submit

Kirimkan melalui link berikut sebelum deadline: https://forms.gle/5C2CR8Nj16Zw1kEa9

Data yang dikirim:

- Link Git repo public
- Link aplikasi terdeploy
- Catatan singkat (opsional): stack yang dipakai, catatan performa, asumsi/limitasi