**Detail Alur Data & Proses Aplikasi CBT**

Dokumen ini menjelaskan secara rinci proses teknis dan aliran data yang terjadi di setiap fitur utama aplikasi, baik dari sisi Admin maupun Siswa.

**Alur Kerja Admin (Rincian Teknis)**

**1. Proses Login**

1. **Input:** Admin memasukkan username dan password di login.php.
2. **Proses PHP:**
   * Data dikirim via metode POST.
   * PHP mencari username di tabel **users**.
   * Jika ditemukan, PHP mengambil password yang ter-hash dari database.
   * Fungsi password\_verify() membandingkan password yang diinput dengan hash dari database.
   * Jika cocok, PHP memulai sesi (session\_start()) dan menyimpan informasi penting seperti $\_SESSION['user\_id'], $\_SESSION['username'], dan $\_SESSION['role'] = 'admin'.
3. **Output:** Admin diarahkan ke admin/index.php.

**2. Proses Manajemen (Ujian, Soal, Siswa)**

Fitur ini menggunakan **AJAX (Asynchronous JavaScript and XML)** secara ekstensif agar admin tidak perlu me-refresh halaman. Alurnya selalu sama:

1. **Memuat Daftar (Tabel):**
   * Saat halaman (misal: manage\_tests.php) dimuat, JavaScript langsung mengirim permintaan **AJAX** ke file itu sendiri dengan parameter ?fetch\_list=true.
   * Di server, blok PHP paling atas mendeteksi parameter ini. Ia melakukan query ke database (misal: SELECT \* FROM tests ...) dengan **filter, pencarian, dan paginasi (LIMIT, OFFSET)**.
   * PHP mengembalikan data dalam format **JSON**.
   * JavaScript menerima data JSON ini, lalu secara dinamis membuat baris-baris tabel (<tr>) dan tombol-tombol paginasi di dalam HTML.
2. **Menambah Data Baru:**
   * Admin menekan tombol "Tambah", JavaScript membuka **modal (popup)**.
   * Admin mengisi form dan menekan "Simpan".
   * JavaScript mengumpulkan semua data dari form (termasuk file gambar/audio jika ada) dan mengirimkannya via **AJAX** ke file yang sama dengan parameter action=add\_....
   * PHP mendeteksi action ini, melakukan validasi, meng-hash password (jika untuk siswa), memproses upload file, lalu melakukan INSERT data ke tabel yang sesuai.
   * PHP mengembalikan status sukses/gagal dalam format **JSON**.
   * JavaScript menerima status ini, menutup modal, dan **memanggil ulang fungsi untuk memuat daftar (langkah 1)** agar tabel otomatis ter-update dengan data baru.
3. **Mengedit Data:**
   * Admin menekan ikon "Edit". JavaScript mengambil ID dari data tersebut.
   * JavaScript mengirim **AJAX** ke file "pembantu" (misal: get\_test\_details.php?id=...).
   * File PHP ini melakukan SELECT ke database untuk satu baris data, lalu mengembalikannya sebagai **JSON**.
   * JavaScript menerima data detail ini, membuka **modal**, dan mengisi semua field form dengan data tersebut.
   * Proses penyimpanan selanjutnya sama seperti proses "Menambah Data Baru", namun PHP akan melakukan UPDATE bukan INSERT.
4. **Menghapus Data:**
   * Admin menekan ikon "Hapus". JavaScript membuka **modal konfirmasi**.
   * Jika dikonfirmasi, JavaScript mengirim **AJAX** ke file yang sama dengan parameter action=delete\_... dan ID data yang akan dihapus.
   * PHP melakukan DELETE dari tabel yang sesuai.
   * JavaScript me-refresh tabel.

**3. Proses Rakit Ujian (assemble\_test.php)**

1. **Memuat Halaman:** PHP mengambil semua soal yang sudah ada di ujian ini dari tabel **test\_questions** dan menampilkannya di sisi kiri.
2. **Menampilkan Bank Soal:** JavaScript secara otomatis memanggil get\_bank\_questions.php via **AJAX** untuk memuat halaman pertama dari daftar soal yang tersedia di sisi kanan.
3. **Mengatur Urutan & Poin:**
   * Admin menyeret soal di sisi kiri. Library SortableJS menangani antarmuka visualnya.
   * Admin mengubah angka di dalam input poin.
   * Saat admin menekan "Simpan Perubahan", JavaScript membaca urutan elemen dan nilai poin dari setiap soal di sisi kiri.
   * Data ini (sebuah array berisi ID soal dan poinnya) dikirim via **AJAX** ke update\_test\_questions.php.
   * PHP melakukan UPDATE pada setiap baris di tabel **test\_questions**, memperbarui kolom question\_order dan points.

**Alur Kerja Siswa (Rincian Teknis)**

**1. Proses Memulai Ujian**

1. **Dasbor:** Saat student/index.php dimuat, PHP melakukan query LEFT JOIN dari tabel **tests** ke **test\_results** dengan student\_id yang sedang login. Ini memungkinkan sistem mengetahui status setiap ujian (sudah dikerjakan, sedang dikerjakan, atau belum).
2. **Konfirmasi:** Saat siswa menekan "Mulai Ujian", mereka diarahkan ke confirm\_page.php.
3. **Mencatat Sesi:** Saat tombol "Mulai Ujian Sekarang" ditekan, PHP melakukan INSERT satu baris baru ke tabel **test\_results**. Baris ini berisi student\_id, test\_id, start\_time (waktu saat itu), dan status diatur ke 'in\_progress'. ID dari baris inilah yang menjadi test\_result\_id.

**2. Proses Pengerjaan Ujian (test\_page.php)**

1. **Memuat Soal:** PHP mengambil test\_result\_id dari sesi ujian yang sedang 'in\_progress'. Kemudian, ia mengambil semua soal dari **test\_questions** yang terkait dengan ujian tersebut, diurutkan berdasarkan question\_order.
2. **Timer:** PHP menghitung selisih waktu antara start\_time dan waktu selesai yang diharapkan (start\_time + durasi). Selisih ini (dalam detik) dikirim ke JavaScript.
3. **Interaksi:** JavaScript menangani semua interaksi: menampilkan soal satu per satu, menandai navigasi soal yang sudah dijawab, dan menjalankan timer.

**3. Proses Selesai Ujian (submit\_test.php)**

Ini adalah proses backend yang paling krusial.

1. **Menerima Data:** PHP menerima test\_result\_id dan sebuah array answers yang berisi [question\_id => student\_answer].
2. **Mengambil Kunci Jawaban:** PHP melakukan satu query besar untuk mengambil semua kunci jawaban (correct\_answer) dan poin (points) untuk semua soal yang ada di ujian tersebut, menyimpannya dalam sebuah array asosiatif.
3. **Kalkulasi & Penyimpanan Jawaban:**
   * PHP melakukan *looping* melalui array kunci jawaban.
   * Di setiap iterasi, ia membandingkan kunci jawaban dengan jawaban siswa.
   * Jika benar, ia menambahkan nilai points dari soal tersebut ke total skor.
   * PHP kemudian melakukan INSERT ke tabel **student\_answers**, mencatat test\_result\_id, question\_id, jawaban siswa, dan status is\_correct (1 atau 0).
4. **Menyimpan Skor Akhir:** Setelah loop selesai, PHP melakukan UPDATE pada baris di tabel **test\_results** yang sesuai. Ia mengisi kolom score dengan total skor yang telah dihitung, mengisi end\_time, dan mengubah status menjadi 'completed'.
5. **Output:** Siswa diarahkan ke result\_page.php dengan result\_id.