**Investigate “View results” shows blank results page**

**Context (do not change logic):**  
This is a swim team optimisation app. The optimisation engine works and must **not** be modified. Results are successfully saved in the DB table optimization\_results. The bug is UI/flow/state (or possibly the read side of the backend), not the optimisation logic.

**Bug summary:**  
From Home → for a team with status **Selected**, clicking **“View results”** navigates to the Results page for that team, but the page renders **blank content** (only stage headers + “Back to home”). When the user resumes an **in-progress** team via **“Continue selection”**, everything restores correctly (including data). So:

* “Continue selection” → good (state restores)
* “View results” (selected team) → navigates to Results but **no swimmer/event rows are shown**
* The DB table optimization\_results contains the expected saved records for that team.

**Goal:**  
Thoroughly investigate and explain why the Results page does not render previously saved results when entered via **“View results”**. Propose targeted fixes. **Do not change any optimisation logic** (e.g., scoring/assignment code, solver config, objective functions). Only inspect/adjust state hydration, routing, and read-side fetch of saved results.

**Reproduction steps (run the app and capture logs)**

1. Start the app locally in Replit. Open logs for frontend and backend.
2. Look at team titled ‘test ages’.
3. Click **“View results”** for that team. Observe: Results page shows stage headers but **no swimmers/events grid**.
4. While reproducing, open DevTools Network tab and backend logs. Capture the **exact requests** and responses when landing on Results via “View results”. Note any 4xx/5xx or empty payloads.

Record:

* The route you land on (e.g., /results/:teamId or query string).
* Whether a **fetch** to read previous results is made on page mount.
* Whether the component tries to read results only from in-memory state/store (which would be empty on cold navigation).
* Any React errors (key warnings, null deref), unhandled promises, or silent conditional rendering (e.g., guards that hide UI when results.length === 0).

**Code paths to inspect (search the repo)**

Please search for these strings and open the closest matches:

* **Frontend components/pages:**
  + Results, ResultsPage, ResultsView, AssignedSwimmers, EventAssignments
  + Buttons: Continue selection, View results
  + Route/Router config: results, teamId, selected, inProgress
  + State management: store, context, redux, zustand, provider
  + Data hooks: useEffect, useQuery, fetchResults, getOptimizationResults
* **Backend read endpoints (DO NOT TOUCH optimisation logic):**
  + Controllers/routers with paths like:
    - GET /api/teams/:teamId/results
    - GET /api/optimization\_results?teamId=...&latest=true
    - Any handler that queries the optimization\_results table
  + Confirm query joins/filters: team\_id, status = 'SELECTED', latest, created\_at DESC LIMIT 1
* **Persistence layer:**
  + Model/entity for optimization\_results
  + Serializer/DTO types sent to the frontend
  + Any code that writes results after optimisation (to confirm shape vs. read shape)

**Likely root causes to check (and what to look for)**

1. **Results page relies only on transient client state**
   * Symptom: When entering via a linear flow (same session), results render; on fresh navigation (“View results”), state is empty so renders nothing.
   * Evidence: No network request on mount; code reads from context/store like const results = useStore(s => s.results) with no fallback fetch.
   * Fix (front-end): On ResultsPage mount, if results absent/empty, **fetch by teamId** and hydrate state. Example:
   * // PSEUDO (React)
   * const { teamId } = useParams();
   * const [data, setData] = useState(null);
   * useEffect(() => {
   * let cancelled = false;
   * async function load() {
   * const res = await fetch(`/api/teams/${teamId}/results?latest=true`);
   * const json = await res.json();
   * if (!cancelled) setData(json);
   * }
   * load();
   * return () => { cancelled = true; };
   * }, [teamId]);
   * Also audit conditional rendering: if data is null, show a loading state; if data is [], show “No results saved”, not a blank screen.
2. **Missing/incorrect teamId in route or fetch**
   * Symptom: Navigation to Results occurs without a valid teamId (e.g., route is /results instead of /results/:teamId), or query param mismatch (team\_id vs teamId).
   * Evidence: Router shows path without param; API call uses undefined in URL; backend log shows no matching team.
   * Fix: Ensure “View results” button links with correct param:
   * <Link to={`/results/${team.id}`}>View results</Link>
   * // or Next.js:
   * router.push(`/results/${team.id}`);
3. **Backend read endpoint returns empty due to filter mismatch**
   * Symptom: API responds 200 with [] or { results: [] } when called via “View results”, but returns data via linear flow.
   * Evidence: DB contains rows for optimization\_results with correct team\_id, but the endpoint filters by wrong status, version, or expects a run\_id only set during fresh optimisation.
   * Fix: Add a **“fetch latest by team\_id”** path that ignores transient run\_id:
   * SELECT \* FROM optimization\_results
   * WHERE team\_id = $1
   * ORDER BY created\_at DESC
   * LIMIT 1;

Ensure the controller maps this to a stable endpoint like GET /api/teams/:teamId/results?latest=true.

1. **Shape mismatch between save and read**
   * Symptom: Backend returns {assignment: ...} but frontend expects {results: ...} (or different keys), leading to guarded render that hides UI.
   * Evidence: Console shows Cannot read properties of undefined (reading 'events').
   * Fix: Align contract or add a mapper on the client before render. Document the expected schema (events[], swimmers[], assignments[]).
2. **Guard/feature flag blocking render for “completed” teams**
   * Symptom: Conditional like if (!canEdit) return null inadvertently hides the results table for selected/locked teams.
   * Fix: Separate “editability” from “visibility”. Results should render in read-only mode when team is selected.
3. **Stale cache or query key** (if using React Query/TanStack)
   * Symptom: Query key omits teamId, so cache returns stale/irrelevant empty data.
   * Fix:
   * useQuery({
   * queryKey: ['results', teamId, 'latest'],
   * queryFn: () => fetch(`/api/teams/${teamId}/results?latest=true`).then(r => r.json())
   * });

**What I want from you before making changes**

1. **Root Cause Analysis**
   * Identify the exact cause(s) with file paths and line references.
   * Show the failing code snippet(s) and explain why they fail on “View results” but not on “Continue selection”.
2. **Proposed Fix (no code changes yet)**
   * Provide a **minimal diff** (as a patch or inline snippet) limited to **frontend hydration/routing** and/or **backend read endpoint**.
   * Explicitly confirm that no optimisation logic or solver code is touched (list the dirs/files you will avoid, e.g., /server/optimizer/\*\*, /lib/solver/\*\*).
3. **Data contract**
   * Document the response shape the Results page expects (keys, types). If mismatched, propose a mapping layer (client or controller) with examples.
4. **Verification Plan**
   * Exact steps to verify fix:
     + Fresh page load via “View results” shows latest saved results from optimization\_results.
     + “Continue selection” still works.
     + Export to XLSX still works and matches UI.
     + Hard refresh on Results page keeps data (no reliance on prior in-memory state).
   * Include a simple integration test or manual checklist.
5. **Risk & Rollback**
   * Note any cached state implications, loading states, and edge cases (no results yet, multiple result runs).
   * Provide a quick rollback plan (revert diff only in UI/read endpoint).

**Constraints:**

* ✅ You may add/adjust client-side data fetching on Results mount and/or add a **read-only “latest results by team\_id”** endpoint (if missing).
* ❌ Do **not** modify any files that implement the optimisation logic, scoring, constraints, or solver configuration.
* ❌ Do not alter how results are **written** post-optimisation (only how they are **read/hydrated** when landing via “View results”).
* 🚫 Do not push changes yet — first, present RCA and proposed diffs for approval.

**Nice to have (optional):**

* Add a small “Loading results…” and “No results found for this team” UI state to avoid blank screens.
* Ensure route guards don’t hide results in read-only mode for selected teams.

**Deliverables:**

* A short report with:
  + Repro evidence (network logs + code refs)
  + Root cause
  + Proposed minimal diff (frontend and/or read endpoint)
  + Verification checklist
  + Confirmation that optimisation logic remains untouched