ULTRA — UI Smoke Test Plan (Post-RLS) (2026-02-09)

Use this checklist to confirm the app is healthy after Step 4 (Permissions & RLS) and the `api/invite-user.ts` rewrite.

This test plan assumes the current (transitional) table names in the live DB:

- Membership: `public.user_group_links`
- Yacht ownership: `public.yacht_group_links` (with `unique(yacht_id)` enforced)
- Templates: `public.tasks`
- Categories: `public.task_categories`
- Instances/assignment: `public.task_contexts`, `public.task_context_assignees`, `public.yacht_tasks`
- Execution history: `public.task_results`, `public.yacht_task_results`

Authoritative intent references:

- `docs/RLS_DESIGN.md`
- `docs/BACKEND_DECISIONS.md`
- `docs/SCHEMA_DECISIONS.md`
- `docs/TERMINOLOGY.md`

Personas to test with

Persona A — Admin operator

An authenticated user who:

- has role `admin` in `public.user_role_links → public.roles`
- is a member of at least one "real" operational group (e.g. Worthy Marine)
- is also a member of **Global Library** (we added all users)

Example: `grichfitz@hotmail.com` (once linked to the `admin` role).

Persona B — Normal user (Global-only)

An authenticated user who:

- is a member of **Global Library** only
- is NOT a member of Worthy Marine (or other operational groups)

This user should see the **global task library**, but typically see **no yachts** (unless they're added to a yacht-owning group).

Persona C — Group member (Worthy Marine)

An authenticated user who:

- is a member of **Worthy Marine** (and optionally Global)

This user should see Worthy Marine yachts + tasks scoped to that group tree, plus global templates/categories.

Test 1 — Authentication and basic navigation

For Personas A/B/C:

- Sign in successfully.
- Load the desktop/home view.
- Open **Tasks** app.
- Open **Yachts** app.
- Open **Users** app (if enabled in navigation).

Expected:

- No infinite spinners.
- No "permission denied" console spam on basic reads.

Test 2 — Group visibility is scoped (RLS)

Persona B (Global-only)

- Go to any group picker/list screens (if present).

Expected:

- Only **Global Library** (and its descendants, if any) are visible.
- Worthy Marine / Dockers / etc. should NOT be visible unless the user is a member.

Persona C (Worthy Marine member)

Expected:

- Worthy Marine visible.
- Any descendant groups visible (if you use parent/child groups).
- Unrelated groups not visible.

Test 3 — Global task library works (templates + categories)

Persona B (Global-only)

- Open Tasks list/tree.
- Confirm templates/categories load.

Expected:

- Global templates/categories are visible.

Persona C (Worthy Marine member)

Expected:

- Global templates/categories visible.
- Any Worthy Marine–private categories/templates visible (if they exist and are pinned to Worthy Marine).

Test 4 — Creating templates (known limitation)

For any persona:

- Try **New Task** (template create).

Expected (current design):

- **May fail with 403** under RLS because `public.tasks` has no safe group anchor for client-side INSERT.
- This is expected until a deliberate server-side create+scope operation (RPC/endpoint) is implemented.

Pass criteria:

- Failure is a clean error message (not silent data corruption).

Test 5 — Creating categories (group-scoped write)

For Persona B (Global-only):

- Try **New Category**.

Expected:

- Category creation succeeds into **Global Library** scope (because the user can see that group).

For Persona C (Worthy Marine member):

- Create a category while in Worthy Marine context (if UI supports choosing).

Expected:

- Category creation succeeds only in groups the user can see.

Test 6 — Yacht visibility and ownership (one group per yacht)

Persona B (Global-only)

- Open Yachts app.

Expected:

- Likely **no yachts** visible (unless yachts are owned by Global Library, which they should not be).

Persona C (Worthy Marine member)

- Open Yachts app.

Expected:

- Worthy Marine yachts visible (owned by Worthy Marine).

Uniqueness check via UI

- Attempt to assign the same yacht to a *second* group (if you have a yacht↔group assignment screen).

Expected:

- The operation should fail due to the `unique(yacht_id)` constraint on `public.yacht_group_links`.
- UI should show a clear error (constraint violation), and yacht should remain in exactly one group.

Test 7 — Assign/unassign templates to a yacht (instance creation via task_contexts)

Precondition:

- Use Persona C and choose a yacht that is visible (e.g., Sea Venture).

Steps:

- Open the yacht task assignment screen.
- Toggle a task ON (assign).

Expected:

- Assign creates `task_contexts` rows (and/or `yacht_tasks` depending on UI), within scope.
- No 403 errors.

Steps:

- Toggle a task OFF (unassign) for a task with **no history**.

Expected:

- Unassign succeeds (deletes `task_contexts` rows).

Test 8 — Execution history blocks unassign (append-only results)

Precondition:

- Pick a yacht + assigned task that has a `task_contexts` row.
- Insert a `task_results` record for that context (you can do this via SQL editor for testing).
    - `task_results.result_status` must be either `pass` or `fail`.

Steps:

- In the UI, try to unassign that task from the yacht.

Expected:

- UI blocks unassign with the message:
    - “This task cannot be unassigned because there is execution history (task_results) for this yacht.”

Test 9 — User invitations (backend + authz)

Persona A (Admin)

- Go to **New User**.
- Select a target group.

- Invite a new email.

Expected:

- Request succeeds.
- Server returns a user id.
- Membership row is created for the invited user in the selected group.

Persona B/C (Non-admin)

- Attempt the same invite flow.

Expected:

- Request fails with 403 (“admin role required”) or 401 if not signed in.

Test 10 — Membership writes are blocked from the client (by design)

For any non-service client UI:

- Try to “assign user to group” (any UI that writes `user_group_links`).

Expected:

- 403 Forbidden (RLS blocks membership writes).
- This is intentional per `docs/RLS_DESIGN.md` (membership is service/admin only).

If a test fails

Capture:

- Which persona (A/B/C)
- Screen/route
- The exact error (console + network request)
- The table being written/read (from the request URL like `/rest/v1/<table>`)

Then we adjust **only the specific single-table policy** (or add an explicitly approved admin backend path), without inventing new permission rules.