WinBackk Audit Reporting System

Developer Handover Brief

Goal

Build an internal WinBackk web app (staff-only) that stores audit data and generates a **professional PDF report** that WinBackk can send to client companies. Clients do **not** log in.

This is a **report generation system**, not a portal/dashboard for clients.

1) What the system must do (plain English)

WinBackk staff will:

1. Create an organisation (client) and site (location)
2. Create an audit (date + notes)
3. Enter exposure map scores (Role × Driver, score 0–5 + evidence)
4. Create controls (actions) linked to the audit (status + due dates + evidence)
5. Click **Generate PDF Report**
6. The system creates a PDF on the server, saves it privately, and gives WinBackk:
    - a **download link** (signed, time-limited), and/or
    - a file to attach to an email

2) Tech stack (locked)

- Frontend: **Next.js App Router**
- Styling: Tailwind
- UI: shadcn/ui
- Auth: **Clerk** (WinBackk staff only)
- DB: **Supabase Postgres**
- Security: **Supabase RLS**
- Storage: **Supabase Storage (private buckets)**
- Charts (optional in UI): Recharts
- PDF generation: **Server-side** (HTML → PDF via headless browser)

3) Users and permissions

Roles (only internal)

- `winbackk_admin` — full access
- `winbackk_staff` — can access only assigned organisations

Multi-tenant rule (critical)

Every table has `org_id`.
RLS must ensure:

- Staff can only read/write data for orgs they are assigned to
- Admin can access all orgs
- No cross-org data leaks

Clerk user metadata example:

```
{
  "role": "winbackk_staff"
}
```

Staff org access is stored in DB table: `org_user_assignments`.

4) Database tables (minimum needed)

(Use the schema you already wrote; these are the ones used in report calculations)

- `orgs`
- `org_sites`
- `org_roles`
- `audits`
- `exposure_drivers` (global reference)
- `exposure_map_rows` (core scores + evidence)

- `mri_snapshots` (stored calculated outputs)
- `controls`
- `documents` (stores report PDFs and other files)
- `audit_log` (mandatory: record changes
-

5) Report types to generate

A) Workplace Movement Risk Audit Report (PDF) — main one

Sections:

- Cover page
- Executive summary
- MRI (overall + by role + change vs previous)
- Exposure map heatmap summary (table/visual)
- Top exposure drivers
- Controls (what's being done)
- Progress summary
- Appendix (scoring scale, method)

6) Exact report calculations (define once, always consistent)

6.1 Definitions

- Scores are stored in `exposure_map_rows.score` as **integer 0–5**
- Each exposure map row is one:
    - `audit_id`, `org_role_id`, `driver_id`, `score`

Assume **higher score = higher risk**.

6.2 Role-level score

For each role in an audit:

Let `S(role)` = average score across drivers for that role.

Formula:

- Collect all rows where:
    - `audit_id = current audit`
    - `org_role_id = this role`
- Then:
    - `S(role) = AVG(score)`

This gives a value from **0.0 to 5.0**.

Also compute a percentage form for readability:

- `S_pct(role) = (S(role) / 5) * 100`

6.3 Driver-level score

For each driver in an audit:

Let `D(driver)` = average score across roles for that driver.

Formula:

- Collect all rows where:
    - `audit_id = current audit`
    - `driver_id = this driver`
- Then:
    - `D(driver) = AVG(score)`
- Percentage:
    - `D_pct(driver) = (D(driver) / 5) * 100`

6.4 Overall MRI score (single number)

Overall MRI should represent the audit's overall risk level.

Use the simplest defensible method:

Method: average of all exposure_map_rows scores for the audit.

Let `MRI_raw`:

- `MRI_raw = AVG(score) over all exposure_map_rows where audit_id = current audit`

Then:

- `MRI_pct = (MRI_raw / 5) * 100` → scale 0–100

Store in `mri_snapshots.overall_score` as **0–100** (rounded to 1 decimal).

6.5 MRI by role (for report table)

Create JSON stored in `mri_snapshots.by_role_json`:

Example:

```
[
  { "role_id": "uuid", "role_name": "Legal Assistant", "raw": 3.6,
"pct": 72.0 },
  { "role_id": "uuid", "role_name": "Partner", "raw": 2.8, "pct":
56.0 }
]
```

Sort roles by highest `pct` first.

6.6 MRI delta vs previous audit

To compute change vs previous audit at the same site/org:

1. Find previous audit:
    - same `org_id`
    - (optional but recommended) same `site_id`
    - `audit_date < current audit_date`
    - order by `audit_date DESC`, take 1
2. Delta:

- `delta_points = current.MRI_pct - previous.MRI_pct`

Show in report as:

- `+X.X points` or `-X.X points`
- If no previous audit exists, show “N/A (first audit)”.

6.7 Top exposure driver

From `D_pct(driver)` above, pick the max:

- `top_driver = driver with highest D_pct`

Report it as:

- Driver name + percentage

6.8 Controls KPIs (for report summary)

Controls live in `controls` table.

For a given audit:

- `open_controls = count where status IN ('todo','doing')`
- `done_controls = count where status = 'done'`
- `overdue_controls = count where status IN ('todo','doing') AND due_date < today`

Also include:

- `completion_rate = done_controls / total_controls * 100` (if total > 0)

6.9 Status change logging (audit trail)

Any update to:

- `controls.status`
- `controls.due_date`
- `controls.success_metric`
- `exposure_map_rows.score`
- evidence fields
  must insert a row into `audit_log`.

Minimum fields:

- `org_id`, `user_id`, `action`, `table_name`, `record_id`, `timestamp`

7) Server-side PDF generation flow (step-by-step)

Overview

PDF must be created on the server (not in the browser) so it's consistent and secure.

Recommended approach

1. Next.js Route Handler: `POST /api/reports/generate`
2. Server loads all needed data from Supabase (scoped by RLS)
3. Server renders a HTML report template
4. Server converts HTML → PDF using headless Chromium (Playwright or Puppeteer)
5. Server uploads PDF to Supabase Storage (private)
6. Server writes a `documents` row with metadata (org_id, audit_id, version)
7. Server returns a **signed URL** for download

Detailed flow

Step 1 — Client action

In internal UI, staff clicks:

- "Generate Report" (selects audit + report type)

UI sends:

```
{
  "org_id": "...",
  "audit_id": "...",
  "report_type": "audit_report"
```

```
}
```

Step 2 — API route validates user

- Read Clerk session
- Get user id + role
- Confirm user has access (RLS should enforce, but still check basic conditions)

Step 3 — Fetch report data (server)

Fetch:

- org + site + audit details
- exposure map rows (with role + driver names)
- controls for audit
- previous audit MRI snapshot (if exists)
- evidence file references (do NOT expose public links)

Step 4 — Compute metrics (server)

Compute:

- `MRI_raw`, `MRI_pct`
- per-role scores and per-driver scores
- delta vs previous audit
- control KPIs
  Then insert into `mri_snapshots` (or update if regenerating).

Step 5 — Build HTML report

Generate a full HTML string from a template:

- Cover page
- Tables for role scores + driver scores

- A “heatmap-like” table (simple and print-friendly)
- Controls table (status, due dates, owner, success metric)
- Notes and evidence summaries

Important: keep styling print-safe (no heavy animations).

Step 6 — Convert to PDF (headless browser)

Use Playwright/Puppeteer:

- load HTML
- wait for fonts
- export PDF with A4, margins, page numbers

Step 7 — Save PDF privately

Upload to Supabase Storage private bucket:
Example path:

- `orgs/{org_id}/audits/{audit_id}/reports/audit-report-v{N}.pdf`

Create/Update `documents` row:

- `org_id`
- `category = "report"`
- `title = "Workplace Movement Risk Audit Report"`
- `version = N`
- `file_url = storage path`
- `created_at`

Step 8 — Return signed URL

Generate a signed URL (expiry e.g. 7 days) and return:

```
{
  "document_id": "...",
  "signed_url": "...",
  "expires_in": 604800
}
```

UI shows:

- “Download PDF”
- “Copy link”
- “Regenerate report”

8) MVP pages (internal only)

- `/orgs` — list, create orgs, assign staff
- `/audits` — create audit + enter exposure map rows
- `/controls` — manage controls per audit
- `/reports` — generate and download PDFs
- `/documents` — view stored reports and evidence files

9) Acceptance checklist (what “done” means)

- Staff can create audit data and controls
- Clicking “Generate PDF” produces a professional PDF
- PDF is stored privately and shareable via signed link
- RLS prevents cross-org access
- Audit log records sensitive changes
- MRI calculations match formulas above exactly (0–100 scale)

- Regenerating report increments version and preserves old versions