OKRFLOW • CLIENT BRIEFING • READ-ALOUD SCRIPT

System Walk-Through (Non-Technical, PRD-Aligned)

This document is designed to be read in front of stakeholders. It explains what OKRFlow does, how the user experience maps to the PRD, and where the main building blocks live in the codebase. For deeper engineering detail (API schemas, data model, ops runbooks), see `docs/system-report.html`.

Contents

One-Minute Overview

OKRFlow is an OKR tracking web application that helps an organization create Objectives and Key Results, align them from company → department → team → individual, and track progress with weekly check-ins and dashboards.

It is built as a single full-stack product: the UI and the API live in the same Next.js application. Data is stored in PostgreSQL through Prisma. Authentication supports SSO providers (Google, Slack, Azure AD) and optional password login depending on configuration.

Plain-English architecture: the browser shows pages; pages call `/api/*`; API reads/writes the database; dashboards recalculate progress; optional scheduled jobs send reminders and end-of-cycle scoring.

Where the "app shell" is defined

- Root layout: `app/layout.tsx`
- Global providers: `components/providers.tsx`
- Header navigation: `components/navigation/AppHeader.tsx`
- Protected routing: `middleware.ts`

PRD Alignment (A–F)

The implementation is aligned to the PRD's feature set. Each PRD section maps to a dedicated part of the UI plus supporting API and database logic.

PRD section	What stakeholders see	Where it is implemented

A. OKR Creation & Alignment	Create objectives, add 1–5 key results, attach initiatives/tasks, set owners, goal types (company→individual), cycle dates, and weighting/priority.	UI: `app/(app)/objectives/`, `app/(app)/okrs/`, `components/objectives/` Data model: `prisma/schema.prisma` (Objective: `goalType`, `priority`, `weight`, `progressType`) Business logic: `lib/okr.ts`
B. Tracking & Progress	Update progress, view progress bars per objective, weekly updates, and end-of-cycle scoring (0.0–1.0).	Check-ins UI: `app/(app)/checkins/` Check-in rules: `prisma/schema.prisma` (`@@unique([keyResultId, userId, weekStart])`) Scoring jobs: `app/api/cron/scoring/route.ts`, `lib/jobs.ts`
C. Dashboards	Company dashboard, team views, and “My OKRs” view with quick actions and progress indicators.	Pages: `app/(app)/page.tsx`, `app/(app)/teams/`, `app/(app)/my-okrs/` Widgets: `components/dashboard/Dashboard.tsx`, `components/objectives/progress-chip.tsx`
D. Collaboration & Check-ins	Weekly check-ins with Green/Yellow/Red, comments/discussion, and reminders for updates.	Comments: `app/api/comments`, `components/collaboration/` Check-in summary logic: `lib/checkin-summary.ts` Notifications: `app/api/notifications`, `lib/notifications.ts`
E. Reporting & Analytics	Export reports to PDF/Excel and see trend/heatmap/tree visualizations.	Export API (manager/admin): `app/api/export/route.ts`, `lib/exporters.ts` Export UI: `components/reports/ExportButton.tsx` Analytics: `components/analytics/HeatMap.tsx`, `AlignmentTree.tsx`, `TimelineView.tsx`
F. Admin & Roles	Role-based access (Admin/Manager/Employee), user/team assignment, SSO integrations for login and reminders.	Admin UI: `app/(app)/admin/` Role enforcement: `lib/rbac.ts`, `middleware.ts` SSO: `lib/auth.ts`, per-org provider configs: `lib/idp.ts`, `prisma/schema.prisma` (`IdentityProviderConfig`)

What “real-time visibility” means here: dashboards and lists update quickly through TanStack Query caching and refetching; it is not dependent on WebSocket infrastructure.

User Journey (End-to-End)

1) Login

- Users log in via SSO (Google/Slack/Azure AD) or password login depending on configuration (`lib/auth.ts` and `.env.example`).
- Protected routes are enforced by `middleware.ts`.

2) Create and align OKRs

- Admins define company OKRs for a cycle; managers align team OKRs beneath them; employees create individual OKRs linked to their team's goals.
- Objective hierarchy is expressed via `Objective.parentId` and `Objective.goalType` (see `prisma/schema.prisma`).

3) Track progress weekly

- Users submit weekly check-ins per key result (Green/Yellow/Red), and the system stores one check-in per key result per week.

4) Review dashboards and reports

- Dashboards summarize where the company stands, what is at risk, and what actions are due next.
- Reports export consolidated OKR status to PDF/Excel for leadership review.

Dashboards (Company / Team / Personal)

The PRD goal is that within 30 seconds, a user understands: (1) overall progress, (2) what is at risk, and (3) what they personally must update. The dashboard layout and widgets are designed for that outcome.

Company dashboard

- Overall progress gauge and key highlights (`components/dashboard/Dashboard.tsx`).
- Team heatmap and at-risk list (`components/analytics/HeatMap.tsx`, `components/productivity/AtRiskObjectivesWidget.tsx`).
- Top objectives snapshot with expandable key results (`components/board/ObjectiveCard.tsx`).

Team dashboard

- Team-focused OKRs and progress rollups (routes under `app/(app)/teams/`).

Personal dashboard ("My OKRs")

- Personal OKR list and quick update actions (`app/(app)/my-okrs/`).

Navigation & search (PRD UI requirement)

- Top navigation + tabs: `components/navigation/AppHeader.tsx`
- Search input routes to: `/okrs?search=...`
- Horizontal nav menu: `components/navigation/AppNavigation.tsx`
- Optional sidebar component exists: `components/navigation/AppSidebar.tsx` (not required for PRD and not wired in the default shell)

Check-ins, Comments, Reminders

Weekly check-ins

- Status colors are a first-class concept (`CheckInStatus`: GREEN/YELLOW/RED).
- One check-in per KR per week is enforced by a database uniqueness constraint (see `CheckIn` in `prisma/schema.prisma`).
- Aggregation and summary logic lives in `lib/checkin-summary.ts`.

Comments and discussions

- Comments can attach to an objective or key result (`Comment.objectiveId` / `Comment.keyResultId`).
- API: `app/api/comments`; UI: `components/collaboration/`.

Reminders and scoring automation

- Cron endpoints: `app/api/cron/reminders/route.ts` and `app/api/cron/scoring/route.ts`.
- Authorization: Vercel Cron uses `Authorization: Bearer $CRON_SECRET` (GET); manual calls can use `x-cron-secret` (POST).

Admin, Roles, Teams

The PRD requires Admin/Manager/Employee roles. OKRFlow stores roles in the database and applies them in middleware, API checks, and UI navigation.

- Role enum: `prisma/schema.prisma` (`Role`).
- Role enforcement and helpers: `lib/rbac.ts`, `middleware.ts`.
- Admin UI: `app/(app)/admin/` and admin APIs under `app/api/admin`.
- Teams: `Team` and `TeamMember` tables in `prisma/schema.prisma`; team views under `app/(app)/teams/`.

Exports (PRD reporting): Export is restricted to Admin/Manager at the API level (`app/api/export/route.ts`), matching typical enterprise expectations.

Login & SSO

- Auth routes: `app/(auth)/login`, `app/(auth)/signup`.
- NextAuth configuration and providers: `lib/auth.ts`.
- SSO providers supported via env variables: Google, Slack, Azure AD (`.env.example`).
- Tenant-specific provider settings: `IdentityProviderConfig` in `prisma/schema.prisma` and loader in `lib/idp.ts`.
- Automatic org provisioning on first login: `ensureOrgAndRole` in `lib/auth.ts`.

How it Runs (Local / Production)

Local development

```
npm ci
cp .env.example .env.local
npx prisma generate
npx prisma migrate dev
npm run db:seed
npm run dev
```

Production essentials

- Set `DATABASE_URL`, `NEXTAUTH_URL`, and `NEXTAUTH_SECRET` (see `.env.example`).
- Configure cron secrets for reminders and scoring (`CRON_SECRET`).
- See deployment checklists in `docs/VERCEL_DEPLOYMENT.md`.

Common Questions (Answer Script)

“What exactly was built?”

OKRFlow is a single web application that includes: an OKR UI, a REST API, authentication, a PostgreSQL data model, reporting exports, and automated jobs for reminders/scoring. The platform supports role-based access and multi-tenant organization separation.

“Where does the backend live?”

The backend is implemented as Next.js API routes under `app/api/*`, using Prisma to read/write PostgreSQL. Shared business rules (progress, scoring, check-ins) are in `lib/*`.

“How is alignment handled?”

Objectives link to each other via a parent-child relationship (`Objective.parentId`) and are categorized by `goalType` (company/department/team/individual). Progress rolls up through weighted key results and objective weights.

“How is progress calculated?”

Key results have `current`, `target`, and `weight`; objective progress is a weighted rollup. The exact formula is documented in `docs/API.md` under “Progress Calculation”.

“How do reminders and end-of-cycle scoring run?”

Cron endpoints (`app/api/cron/reminders/route.ts`, `app/api/cron/scoring/route.ts`) trigger jobs in `lib/jobs.ts`. Access is protected by `CRON_SECRET`.

OKRFlow — Read-Aloud Client Briefing • Companion deep report: `docs/system-report.html`