eWorkConnection

Design and Architecture Document

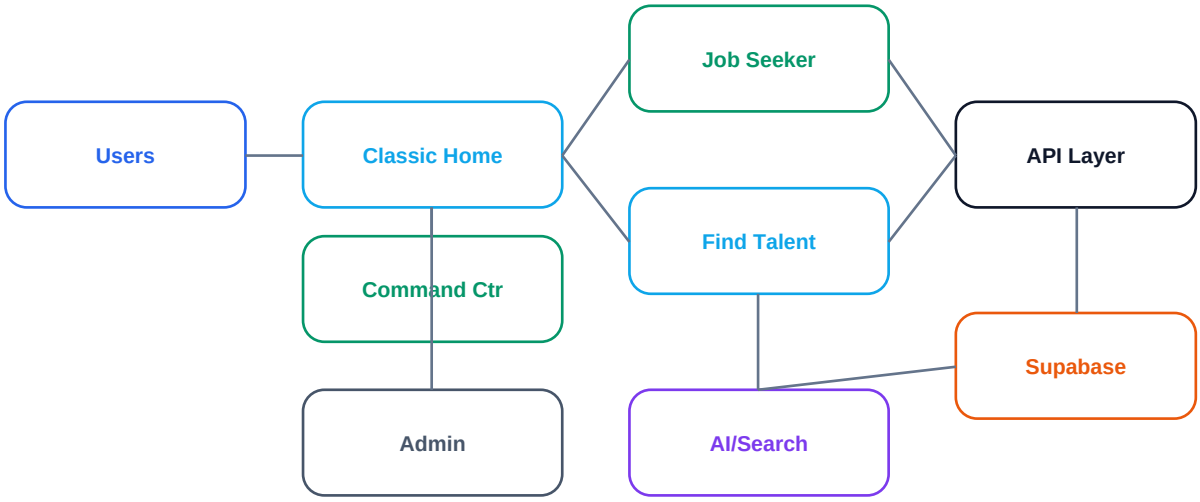
System design, component responsibilities, data architecture, security controls, and launch operations.

Document	Design and Architecture Document
Version	Soft Launch planning set
Date	July 25, 2026
Audience	Founder, admins, developers, QA employee agent, launch partners

Prepared for launch planning. This document is not legal, tax, employment, immigration, financial, or security advice; local counsel and tax advisors should review country-specific launch materials before paid scale.

1. Architecture Overview

The application is built as a Next.js app deployed through Vercel, with Supabase for authentication, database, and row-level security. Server-side API routes mediate AI calls, budget enforcement, tracking, support chat, event logging, and admin actions.



Control points: privacy acknowledgment, role guard, country launch controls, API budget guard, RLS/service role, audit/event logging.

2. Component Map

Layer	Components	Responsibilities
Frontend	Classic Home, Career Command Center, Dashboard, Recruiter, Company Workspace, Admin, Services, Event Log	Role-based UI, alternate entry points, search workflows, filters, consent, application tracking, job posting, employer verification, recruiter network, support and feedback entry.
API Routes	/api/claude, /api/career-briefing, /api/search-cache, /api/recruiter/*, /api/admin/*, /api/support/chat, /api/plans	Server-side AI proxy, cached daily briefing, cost controls, company/public job search, recruiter jobs/workspaces, event ingestion, admin updates, support automation, pricing plan lookup.
Data	Supabase tables and RLS	Profiles, applications, jobs, recruiter job posts, company workspaces, pipeline items/events, user promotions, career briefing cache, usage events, country pricing, peer/recruiter network, event logs, support tickets, budget settings.
Automation	GitHub Actions employee/QA/developer workflows and daily issue summary	Scheduled triage, regression test, P0/P1 summary, email summary, issue creation/handoff, developer fix workflow.
External Services	Anthropic/API key, Resend, Stripe/possible payment providers, public search sources	AI summarization/search generation, email notifications, payments, public data discovery.

3. Data and Regional Design

- Country is detected from profile, browser/launch settings, or user selection and stored for country-specific trial/pricing/access decisions.
- Launch availability is controlled by `country_pricing` settings: country active flag, trial start/end date, free trial days, extension days, and pricing.
- User-specific promotions live separately from country/global coupons and are selected during checkout before country/global promotions.
- Company workspace data is scoped by owner/member records and should never expose another company's internal jobs or pipeline to unrelated users.
- Future regional data residency should separate users by region: North America, Europe/UK, Latin America, India/South Asia, and other expansion zones.
- For EU/UK expansion, data minimization, consent withdrawal, deletion workflows, DPA/vendor review, and cross-border transfer review are required before enabling launch.

4. Security and Privacy Controls

Security is defined as layered controls across identity, authorization, browser exposure, server-side APIs, database policies, cost governance, and operational monitoring. The application should assume that normal users can inspect browser code and direct URLs, so privileged actions must be enforced on the server and in Supabase policies, not only hidden in the UI.

Control	Where Defined	Risk Reduced
Authentication	Supabase OAuth/session and auth callback role routing.	Prevents anonymous access to account-only pages and creates a trusted session boundary.
Authorization	Admin page guard, built-in super-admin list, profile role flags, country admin scopes.	Prevents direct URL access to admin/super-admin functions and limits country-specific control.
Database Access	Supabase RLS policies plus service-role-only server routes for privileged reads/writes.	Prevents client-side users from reading or modifying protected rows outside policy.
Secrets	Vercel environment variables for platform API keys; no platform key exposed to the browser.	Reduces key theft and prevents users from bypassing budget controls.
Same-Origin API	Security helper and API route checks for allowed origins/headers.	Reduces cross-site abuse of paid processing endpoints.
Budget Guard	API usage tables, admin budget route, call-type tracking, maintenance-mode enforcement.	Limits financial exposure from runaway usage, attack traffic, or search loops.
Consent	Service-entry privacy acknowledgment stored in cookie/localStorage; consent withdrawal requirement.	Improves notice/consent posture and reduces repeated user interruption.
Employer Verification	Admin-reviewed recruiter verification fields and trigger/policy restrictions against self-approval.	Prevents unverified recruiters from activating sensitive hiring flows.
Company Workspace	Workspace/member tables, active/verified/public availability filters, and server-side search inclusion rules.	Prevents internal company jobs from leaking to public search.
Auditability	Event log, support tickets, daily P0/P1 summary, user-reported issue metadata, QA artifacts, downloadable diagnostic traces.	Supports defect triage, abuse review, and evidence for operational decisions.

Control	Where Defined	Risk Reduced
UI Safety	Audit/trace UI hidden from normal users; feedback and support placed away from core navigation.	Reduces accidental exposure of internal diagnostics and improves usability.
Plan Isolation	Separate `plan` and `recruiter_plan` columns; Stripe webhook branches writes by plan-key prefix.	Prevents a subscription change on one role from silently overwriting the other role's tier on dual-role accounts.
Admin Bypass Scope	Client bypass in `getEffectivePlan()`/`getEffectiveRecruiterAccess()`; server bypass (`isAdminBypass`) in `/api/claude` usage checks.	Removes plan/trial ceilings for trusted operator accounts without weakening feature-flag rollout gating.
Regulated Section Gate	Global kill-switch flag plus per-account grant (`adult_events_access`, `companion_access`); only super-admin auto-unlocks Adult Events.	Prevents a global rollout flag alone from exposing a regulated section to every admin or user.

5. Security Control Design Principles

- Server enforcement first: UI hiding is only a usability layer; API routes and database policies must enforce the real boundary.
- Least privilege: normal users get service flows, admins get operational controls for assigned countries, and super admins own sensitive platform-wide switches.
- No browser secrets: platform AI/API keys stay in Vercel and are called through server routes with budget checks.
- Data minimization: store only what is needed for account, search, consent, tracking, and support workflows; public data should be labeled and verified by users.
- Auditable change path: feature flags, budget updates, trial resets, user role changes, and maintenance toggles should leave an event trail.
- Country-aware compliance: country launch state, trial, pricing, and data handling should be configurable before the app is opened in that market.

6. AI and Search Flow

- The user initiates a search; the app checks launch, role, budget, maintenance, and enabled sources.
- The server API route calls the AI provider using the platform key unless a user explicitly brings their own key.
- Results are normalized into job/candidate records and stored or displayed with source, freshness, confidence, and validation messaging.
- Subsequent filter/sort actions run client-side against loaded results to reduce cost and latency.
- Completed job and talent searches are stored as browser-local search runs for guest/trial usability; signed-in server records remain the authoritative database-backed source for persisted account analytics.
- Career Command Center merges server counts with browser-local search-run counts so immediately completed searches are visible even before database persistence catches up.
- Job link validation performs deterministic URL checks, server-side page checks, closed-posting phrase checks, and AI active-posting review before results are shown.
- If exact results are empty, fallback/similar suggestions are generated where appropriate, and ordinary zero-result cases should not be logged as defects.

7. Operations Model

Operational Area	Process
Release	Developer commits to GitHub; Vercel deploys; QA agent validates public/auth/admin flows.
Employee Agent	Runs 7 AM and 5 PM America/Chicago; reviews event log, runs regression, sends email summary, and creates actionable issues.
Daily Issue Summary	Runs once daily; sends only P0/P1 event-log and support-ticket items, respecting the Issue reporting email feature flag.
Developer Agent	Receives approved issues or handoff items, prepares fixes, runs tests, and submits for admin/super-admin approval before production.
Incident	Budget breach, auth failure, search failure, or high-severity event triggers maintenance for processing and admin email notification.
Country Expansion	Enable country in admin, set trial/pricing, review legal/privacy/tax, run country QA, then market to target segment.

8. Technical Risks and Mitigations

Risk	Mitigation
AI search cost spikes	Budget guard, call-type tracking, model tiering, caching, filter-only changes, and admin-maintenance fallback.
Public data accuracy	Source labels, verification warnings, confidence scores, and no guarantee language.
Role confusion	Clear role selection, role guard, direct navigation to selected role page, and dual-role support.
Mobile overcrowding	Mobile-first focus selector, collapsible advanced controls, bottom-left feedback, bottom-right chat, feature-flagged app install prompt, and hidden diagnostic bars for normal users.
Compliance drift by country	Country launch controls, local policy review, consent withdrawal, tax/legal sign-off before paid launch.
Cross-role plan collision	Separate `profiles.plan` (job seeker) and `profiles.recruiter_plan` (recruiter) columns; Stripe webhook branches writes by plan-key prefix so a subscription change on one role can never overwrite the other role's tier on a dual-role account.
Regulated feature over-exposure	Two-layer gate for Adult Events/Companion: a global kill-switch flag plus a per-account grant, so raising the global flag alone never exposes the section to unassigned accounts, including admin.

9. Plan and Access Architecture (July 2026)

This section documents the current subscription-plan and privileged-access model, including the changes made to eliminate a cross-role data collision and to give administrative accounts an unconditional, auditable bypass of plan/trial limits.

9.1 Separate Plan Columns

- `profiles.plan` stores the job-seeker subscription tier: free, basic, pro, or expert.
- `profiles.recruiter_plan` stores the recruiter subscription tier: recruiter_basic, recruiter_starter, recruiter_pro, or recruiter_agency, added as its own column with a default of recruiter_basic and a check constraint restricting it to the four

known values.

- Prior architecture stored both tiers in the single `plan` column. On a dual-role account (`has_both_roles = true`), a recruiter subscription renewing or canceling would overwrite the job-seeker plan value in that same column, and a job-seeker subscription change would equally overwrite the recruiter tier. This was a genuine, silent data-collision bug, not a display issue.
- `app/api/webhook/route.ts` now inspects the Stripe subscription's plan metadata key: if it starts with `recruiter_`, the write targets `recruiter_plan`; otherwise it targets `plan`. This branching is applied both when a subscription becomes active and when it is canceled/deleted.
- Client-side read paths (`app/recruiter/page.tsx`) prefer `recruiter_plan` and fall back to the legacy `plan` value only when it happens to already hold a `recruiter_`-prefixed string, covering any account not yet backfilled after the column was introduced.
- The admin Users & Access screen exposes two independent plan `` dropdowns for any account with `has_both_roles` or `recruiter_access`: one bound to `plan`, one bound to `recruiter_plan`, so an operator can no longer accidentally set the wrong role's tier from a single control.

9.2 Admin / Super-Admin Bypass

- Identity for the bypass is resolved from either a hard-coded email allowlist (`BUILT_IN_ADMIN_EMAILS` in `lib/admin.ts`, mirrored locally in each page for client-side gating) or the `profiles.is_admin` / `profiles.is_super_admin` boolean flags.
- Job-seeker side: `getEffectivePlan()` in `app/dashboard/page.tsx` short-circuits to the equivalent of the Expert tier (no trial phase, no day-count limit) at the very top of its logic, before any plan/trial calculation runs, whenever the resolved identity matches an admin or super-admin.
- Recruiter side: `getEffectiveRecruiterAccess()` in `app/recruiter/page.tsx` short-circuits to the Agency tier (every candidate source, unlimited searches, no trial window) using the same identity check.
- Server side: `/api/claude/route.ts` computes `isAdminBypass` from the authenticated user's email/profile flags and skips the usage-limit checks entirely for both `search` and `resume` call types when true — this closes the gap where a determined client could otherwise bypass a client-only check by calling the API route directly.
- The bypass is unconditional with respect to plan and trial state, but it is not a blanket override of the feature-flag rollout system in section 9.3 below: an admin still cannot see a feature whose rollout tier is `off`.

9.3 Rollout-Tier and Regulated-Access Architecture

- `ROLLOUT_CONTROLLED_FLAGS` (currently `career_bot_enabled`, `company_workspace_enabled`, `proactive_briefing_enabled`) each carry a rollout mode of off, super_admin, admin, or normal_users, evaluated by a single `flagAllowed()` helper in `app/api/flags/route.ts` shared by every consumer of the flags API.
- Adult Events adds a second, independent access layer on top of its own flag family: the global `adult_roles` flag is a master kill switch (off hides the section from everyone, including super admin), and `profiles.adult_events_access` is a per-account grant. Super-admin accounts auto-unlock the section without needing the explicit per-account flag; plain admin accounts do not — they must be granted `adult_events_access` individually, and only a super-admin can perform that grant from the admin console.
- Companion Services mirrors this two-layer pattern: `companion_section_enabled` is the global availability toggle, and `profiles.companion_access` is the per-account grant. As of this update, the global toggle is ON in production, but no account (including the operating super-admin account used for testing) currently has `companion_access` granted, so the Companion tab correctly does not appear for anyone yet — this is expected behavior, not a defect.
- Company Workspace uses only the single rollout-tier dial (no separate per-account layer); as of this update it is set to Off, so the feature, its Talent Desk tab, and its admin summary table are all inactive platform-wide.

9.4 Career Bot Architecture

- The alternate homepage `/career-bot-home` (`components/RobotDrivenHome.tsx`) embeds the same mascot/voice widget used elsewhere (`components/CareerBotWidget.tsx`); both route product commands through a shared scripted-intent matcher (`lib/career-bot-intents.ts`) to avoid duplicating common routing rules, while each file intentionally keeps a few local regex branches that must not be blindly merged together.
- Visibility is controlled by two independent flags: `career_bot_enabled` (the rollout tier described above) gates signed-in users, and `career_bot_guests_enabled` (off by default) independently gates anonymous/logged-out visitors on top of that tier.
- Command routing first attempts scripted intent matching; anything unmatched falls through to a conversational AI fallback so general questions are answered rather than silently failing or misrouting. Every reply is labeled Instant or AI Reasoned in the UI so the distinction is visible to the user, not just internal to the code.
- A lightweight per-user memory table (`career_bot_memory`) stores only the last intent title/href and an interaction count, is signed-in only, and is best-effort/fails-silently so a memory error never breaks the chat.
- A guest-accessible single-bullet resume reaction (`/api/resume/react`) is the free teaser: it calls a low-cost model with a rule-based fallback and never persists the pasted text, distinguishing it from the full resume scan/rewrite routes which require sign-in.