Agent Bob — Technical Documentation

Internal reference. Generated 2026-05-29.

1. Overview

Agent Bob is a friendly onboarding tool that helps non-technical teams design a governed AI agent. It walks a user through a 9-step wizard (purpose → personality → tools → autonomy → budget → memory → approvals → review → download) and outputs a deployment-ready 7-file governance package (blueprint, policy, trust manifest, routing rules, run-report template, approval gates, deployment checklist).

It is part of the CEO BOB suite, built by Master 22 Solutions, and is powered by the ClawMaven governance engine.

- Preview: https://id-preview--ad7002e5-0a6c-4b0d-8d08-2ac66f6e7408.lovable.app
- Production: https://agent-bob.lovable.app
- Custom domain: https://agent.ceobob.com

2. Technology Stack

Frontend

- React 19 + TypeScript (strict mode)
- TanStack Start v1 (full-stack React framework, SSR + server functions)
- TanStack Router (file-based routing in `src/routes/`)
- TanStack Query for data fetching
- Vite 7 as build tool
- Tailwind CSS v4 (via `src/styles.css`, native `@import` + theme variables)
- shadcn/ui component library (Radix UI primitives)
- lucide-react icons
- react-hook-form + zod for forms & validation
- sonner for toasts
- recharts for charts

Backend / Runtime

- Cloudflare Workers runtime (edge), deployed via `@cloudflare/vite-plugin` and `wrangler`
- TanStack server functions (`createServerFn`) for app-internal RPC
- Server routes under `src/routes/api/` for webhooks and public APIs
- Lovable Cloud (managed Supabase) for database, auth, and storage
- Supabase Postgres with Row-Level Security

- Stripe (sandbox + live) for payments via embedded checkout
- JSZip for client-side ZIP package generation
- jsPDF + jspdf-autotable for PDF generation (deployment guide)

Governance Integration

- ClawMaven MCP (Model Context Protocol) client — `src/lib/mcp/clawmavenClient.ts`
- Local fallback when governance engine is unreachable

Tooling

- ESLint, Prettier, Bun package manager

3. Application Structure

```
src/
├── routes/                # File-based routes (TanStack Router)
│   ├── __root.tsx          # Root layout (Header/Footer wrapper, <Outlet/>)
│   ├── index.tsx           # Landing page
│   ├── builder.tsx         # Wizard host route
│   ├── review.tsx          # Pre-export review
│   ├── download.tsx        # Final download / deploy package
│   ├── about.tsx           # About (CEO BOB + Master 22 + ClawMaven)
│   ├── pricing.tsx         # Bob Free / Bob Pack / Bob Studio
│   ├── faq.tsx
│   ├── privacy.tsx
│   ├── terms.tsx
│   ├── checkout.return.tsx  # Stripe return page
│   ├── sitemap[.]xml.tsx    # SEO sitemap
│   └── api/
│       ├── mcp.tsx                    # ClawMaven MCP proxy
│       └── public/
│           ├── vertical-packs.ts        # Public vertical pack feed
│           └── payments/webhook.ts      # Stripe webhook
├── components/
│   ├── wizard/             # 7-step wizard components
│   ├── layout/             # Header, Footer, GovernanceStatusPill
│   ├── brand/              # Logo, Disclaimer
│   ├── ui/                # shadcn primitives
│   ├── StripeEmbeddedCheckout.tsx
│   └── PaymentTestModeBanner.tsx
├── lib/
│   ├── wizard/             # Wizard state (Context) + types
│   ├── governance/         # Package builder, risk, approvals, PDF, ZIP
│   ├── mcp/                # ClawMaven MCP client
│   ├── content/            # FAQs
│   ├── stripe.ts           # Browser Stripe loader
│   └── stripe.server.ts     # Server-side Stripe + webhook verify
├── hooks/                 # useGovernanceStatus, useVerticalPacks, ...
├── utils/
│   └── payments.functions.ts  # createCheckoutSession (server fn)
├── integrations/supabase/   # Auto-generated client + middleware
├── styles.css              # Tailwind v4 + design tokens (oklch)
├── start.ts               # TanStack Start instance
└── server.ts               # SSR entry (Worker)
```

4. Core Features

4.1 9-Step Agent Builder Wizard

Lives under `/builder` and is orchestrated by `src/components/wizard/WizardShell.tsx`. State is held in a React Context (`src/lib/wizard/context.tsx`) and persisted to `localStorage` so users can resume.

Step	File	Purpose
1	`Step1Purpose.tsx`	Vertical + preset selection, agent name, purpose, audience, outcome
2	`Step2Personality.tsx`	Personality preset (or custom)
3	`Step3Tools.tsx`	Tool toggles + custom-tool entry
4	`Step4Autonomy.tsx`	Suggest / Draft-for-approval / Act-with-limits / Advanced
5	`Step5Budget.tsx`	Cost ceilings and rate limits
6	`Step6Memory.tsx`	None / Project / Customer / Full memory
7	`Step7Approvals.tsx`	Auto-derived approval rules (editable)
8	`/review`	Risk score + package preview
9	`/download`	ZIP / deploy-ZIP / individual files

4.2 Governance Package Builder

`src/lib/governance/packageBuilder.ts` assembles the 7 files emitted by Bob:

1. blueprint.md — agent design summary
2. policy.yaml — operating policy
3. trust-manifest.json — capabilities, scopes, signing metadata
4. routing-rules.json — tool-call routing
5. run-report-template.md — post-run reporting scaffold
6. approval-gates.yaml — derived from Step 7
7. deployment-checklist.md — go-live checks

Supporting modules:

- `approvalRules.ts` — `deriveApprovalRules(draft)` infers gates from autonomy/budget/memory/tools.
- `riskScoring.ts` — calculates a composite risk score shown on `/review`.
- `verticalPacks.ts` + `agentTemplates.ts` — preset packs per industry vertical.
- `deployPdf.ts` — generates `README-deploy.pdf` via jsPDF.
- `zipExport.ts` — `downloadZip()` (governance package) and `downloadDeployZip()` (package + PDF).

4.3 ClawMaven Vertical Sync

- `src/lib/clawmavenVerticalSync.ts` + `src/hooks/useVerticalPacks.ts` fetch the latest vertical packs from ClawMaven, with a local fallback set in `verticalPacks.ts`.
- `src/components/wizard/SyncBadge.tsx` shows sync status.
- `src/components/layout/GovernanceStatusPill.tsx` + `useGovernanceStatus.ts` show whether the governance engine is connected, checking, or unavailable.

4.4 ClawMaven MCP Bridge

- Server route: src/routes/api/mcp.tsx — proxies MCP requests to ClawMaven.
- Client wrapper: src/lib/mcp/clawmavenClient.ts.

4.5 Payments (Stripe)

- Embedded checkout via @stripe/react-stripe-js — src/components/StripeEmbeddedCheckout.tsx.
- Server function createCheckoutSession in src/utils/payments.functions.ts:
- Validates priceId, agentId, and returnUrl (allow-listed hosts).
- Resolves the Stripe price by lookup_key, decides one-time vs subscription, creates an embedded checkout session.
- Webhook at /api/public/payments/webhook (src/routes/api/public/payments/webhook.ts):
- Verifies signature via verifyWebhook in src/lib/stripe.server.ts.
- Handles checkout.session.completed, customer.subscription.created|updated|deleted.
- Upserts into payments / subscriptions tables using the service-role Supabase client.
- Sandbox/Live split: ?env=sandbox|live query param selects the Stripe environment; client environment is inferred from pk_test_ prefix in VITE_PAYMENTS_CLIENT_TOKEN.
- Pricing tiers (/pricing):
- Bob Free — $0
- Bob Pack — $29 one-time
- Bob Studio — $19 / month
- PaymentTestModeBanner.tsx flags sandbox mode in the UI.

4.6 SEO

- Per-route head() metadata (title, description, OG tags) on every shareable page.
- public/robots.txt, public/llms.txt, and dynamic /sitemap.xml route.
- Semantic HTML, single H1 per page, responsive viewport.

4.7 Design System

- Tokens defined in src/styles.css using oklch() — --background, --foreground, --brand, --brand-soft, --risk-low|med|high, etc.
- Components use semantic Tailwind classes only (no hard-coded colors).
- Dark-mode aware.

5. Data Model (Supabase / Lovable Cloud)

public.payments

One-time Stripe purchases. Keyed by stripe_session_id.

Columns: id, stripe_session_id (unique), stripe_payment_intent_id, customer_email, stripe_customer_id, price_id, product_id, amount, currency, status, agent_id, environment (sandbox/live), created_at.

public.subscriptions

Recurring Stripe subscriptions. Keyed by stripe_subscription_id.

Columns: id, stripe_subscription_id (unique), stripe_customer_id, customer_email, product_id, price_id, status, current_period_start, current_period_end, cancel_at_period_end, environment, created_at, updated_at.

RLS posture

Both tables have RESTRICTIVE deny-all RLS policies for anon and authenticated roles. Reads and writes only happen via the service role from the Stripe webhook handler (which bypasses RLS by design). No end-user CRUD path exists.

6. Server Functions & Routes

Server functions (createServerFn, RPC from the browser)

Function	File	Notes
createCheckoutSession	src/utils/payments.functions.ts	Validates input, creates Stripe embedded checkout session

Server routes (HTTP)

Route	File	Purpose	
GET /api/public/vertical-packs	src/routes/api/public/vertical-packs.ts	Public read-only feed of vertical packs	
`POST /api/public/payments/webhook?env=sandbox	live`	src/routes/api/public/payments/webhook.ts	Stripe webhook (signature-verified)
* /api/mcp	src/routes/api/mcp.tsx	ClawMaven MCP bridge	

SSR / error handling

- src/start.ts registers an errorMiddleware that renders a friendly error page via src/lib/error-page.ts.
- src/server.ts is the Worker entry (referenced from vite.config.ts and wrangler.jsonc).

7. Configuration & Environment

Variable	Scope	Purpose
VITE_SUPABASE_URL	Browser	Supabase project URL

Variable	Scope	Purpose
VITE_SUPABASE_PUBLISHABLE_KEY	Browser	Anon key
VITE_SUPABASE_PROJECT_ID	Browser	Project ref
VITE_PAYMENTS_CLIENT_TOKEN	Browser	Stripe publishable key (test/live drives env)
SUPABASE_URL	Server	Used by service-role client in webhook
SUPABASE_SERVICE_ROLE_KEY	Server	Service-role key (webhook only)
STRIPE_SECRET_KEY / STRIPE_WEBHOOK_SECRET (per env)	Server	Stripe server-side ops

Files like .env, src/integrations/supabase/client.ts, and src/integrations/supabase/types.ts are auto-managed by Lovable Cloud.

8. Deployment

- Built and deployed by Lovable to Cloudflare Workers (edge).
- wrangler.jsonc sets compatibility_date: 2025-09-24 with nodejs_compat flag.
- vite.config.ts uses @lovable.dev/vite-tanstack-config (no manual plugin setup needed).
- Stable URLs:
- project--ad7002e5-0a6c-4b0d-8d08-2ac66f6e7408.lovable.app (prod)
- project--ad7002e5-0a6c-4b0d-8d08-2ac66f6e7408-dev.lovable.app (preview)
- Custom domain: agent.ceobob.com.

9. Security Posture

- RLS enabled on all public tables with explicit deny-all for client roles.
- Stripe webhook signature verification before any DB write.
- createCheckoutSession validates priceId, agentId, and an allow-list of returnUrl hosts (agent-bob.lovable.app, agent.ceobob.com, .lovable.app, .lovable.dev, localhost).
- Service-role key only used server-side; never imported from client code.
- No anonymous sign-ins; no end-user PII stored beyond Stripe customer email on payment rows.
- Security scans run via Lovable's scanner; current state: no active findings.

10. Roadmap Hooks (already wired)

- ClawMaven vertical pack auto-sync (live)
- Governance engine status indicator (live)
- Deployment PDF + ZIP exports (live)
- Stripe sandbox/live dual environment (live)