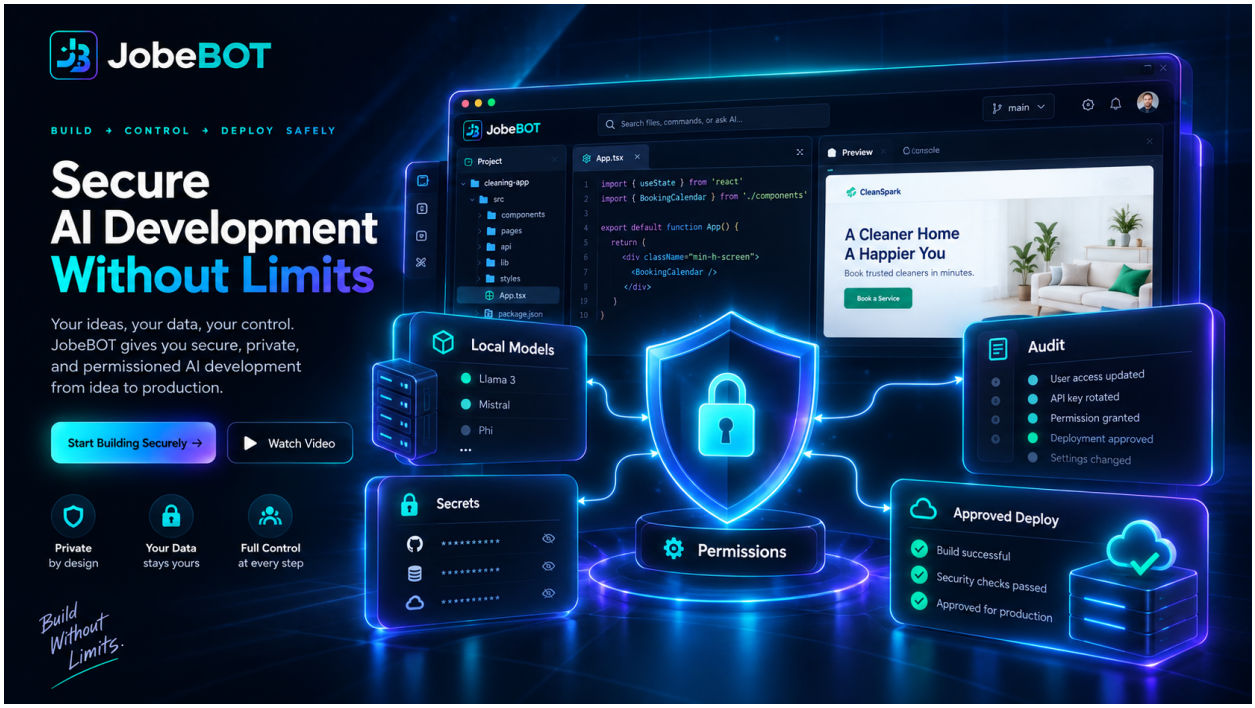
JobeBOT Documentation

Product handbook for JobeBOT Web, JobeBOT WorkSpace, agents, models, and safe project delivery

This handbook explains what JobeBOT does, how its web application and desktop WorkSpace fit together, how agents and models are selected, how files are written and verified, and how teams can use the system responsibly.

At a glance

Surface	Purpose	Typical user action
JobeBOT Web	Account, catalogue, credits, team, settings, downloads, and service APIs.	Sign in, manage access, inspect models, and obtain the desktop application.
JobeBOT WorkSpace	Local coding environment with project files, preview, editor, terminal, agent activity, and evidence.	Open a project, describe the change, approve sensitive actions, and verify the result.
Gateway and bridge	Connect the UI to local or configured provider routes.	Choose a model route and inspect provider health.



JobeBOT is designed around inspect, plan, act, verify, and explain.

1. How JobeBOT works

A user prompt enters the selected surface. JobeBOT establishes project context, identifies the requested outcome, and decides whether the request is conversation, inspection, planning, or implementation. Implementation work is routed through an agent loop that can inspect files, propose changes, call approved tools, write to the project, and verify the result.

• Context: project path, files, local URL, prior run state, and relevant instructions are gathered.

• Intent: the request is classified so a routing or explanatory question does not trigger unnecessary writes.

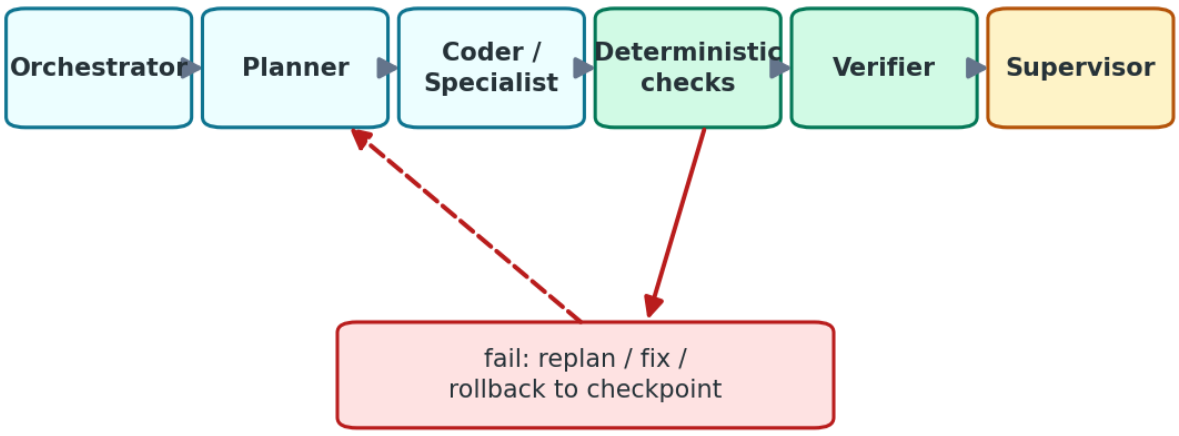
• Plan: the agent identifies files, dependencies, checks, and completion criteria.

• Action: approved tools execute within project scope.

• Evidence: changes are read back, syntax and browser checks run, and the run records its status.

• Completion: JobeBOT reports completed, blocked, or unverified; prose alone is never proof of a file write.

Coder never marks its own work complete — verification is independent



Agent lifecycle: context and intent precede action; verification precedes completion.

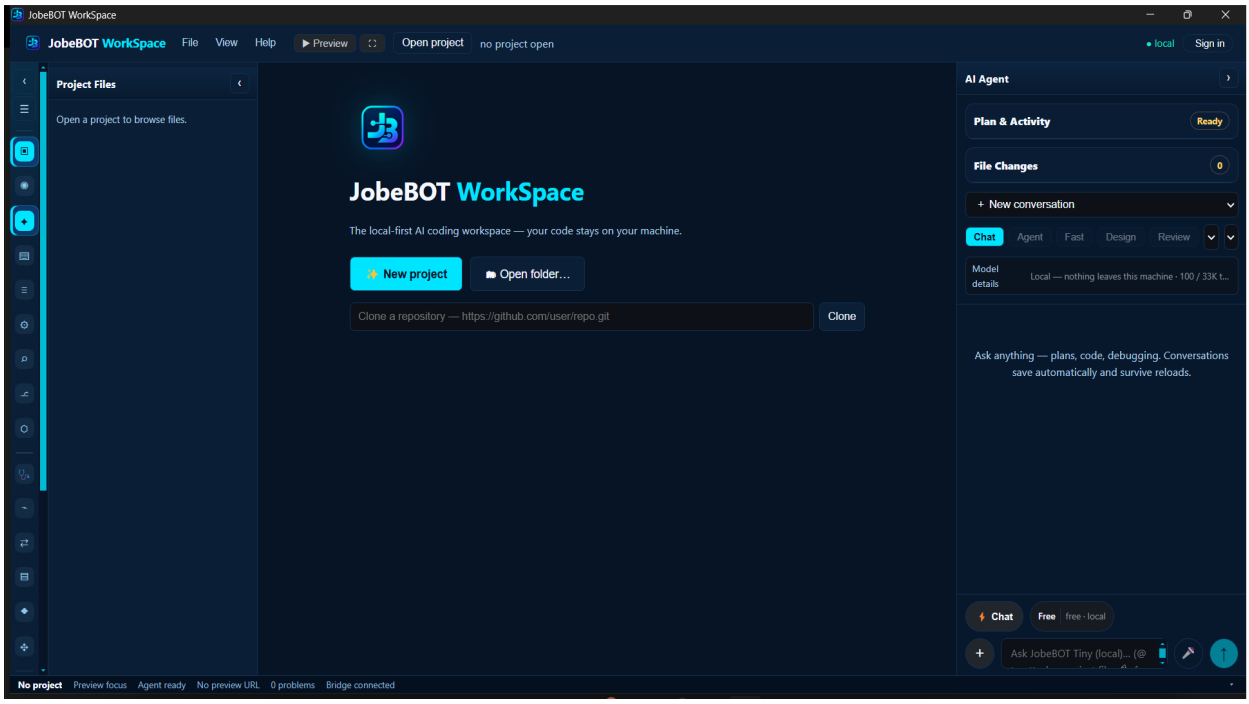
2. JobeBOT Web application

The web application is the account and service control plane. It manages identity, teams, plans, credits, API keys, devices, model catalogue data, and downloads. The desktop WorkSpace is the execution surface for local project work.

Web area	What it does
Home	Explains product capabilities, agents, models, security, pricing, and downloads.
Dashboard	Shows account-level activity and access to project-facing services.
Models	Displays catalogue entries, providers, capabilities, certification state, and route information.
Team	Manages organization membership, invitations, seats, policies, and limits.
Credits and billing	Shows balance, usage controls, plan settings, and billing workflows.
Settings	Manages profile, security, sessions, API keys, webhooks, notifications, and accounts.
Download	Provides desktop application and platform-specific releases.

3. JobeBOT WorkSpace

WorkSpace is a local-first coding environment. The active project path is visible at the top, while the main workspace combines a project file tree, preview or editor, agent activity, and diagnostics. A local project remains on the machine unless the selected provider route or an explicitly approved remote operation sends data elsewhere.



WorkSpace brings project files, preview, agent activity, and verification into one surface.

- Project files: inspect the actual working tree and open files in the editor.
- Preview: load the local URL, refresh, diagnose, inspect ports, and verify the rendered result.
- Editor and Coding: make targeted edits and run checks with file evidence.
- AI Focus: see the current agent, model route, prompt, activity, and completion contract.
- Terminal: run approved commands in the project working directory.

4. Agents and roles

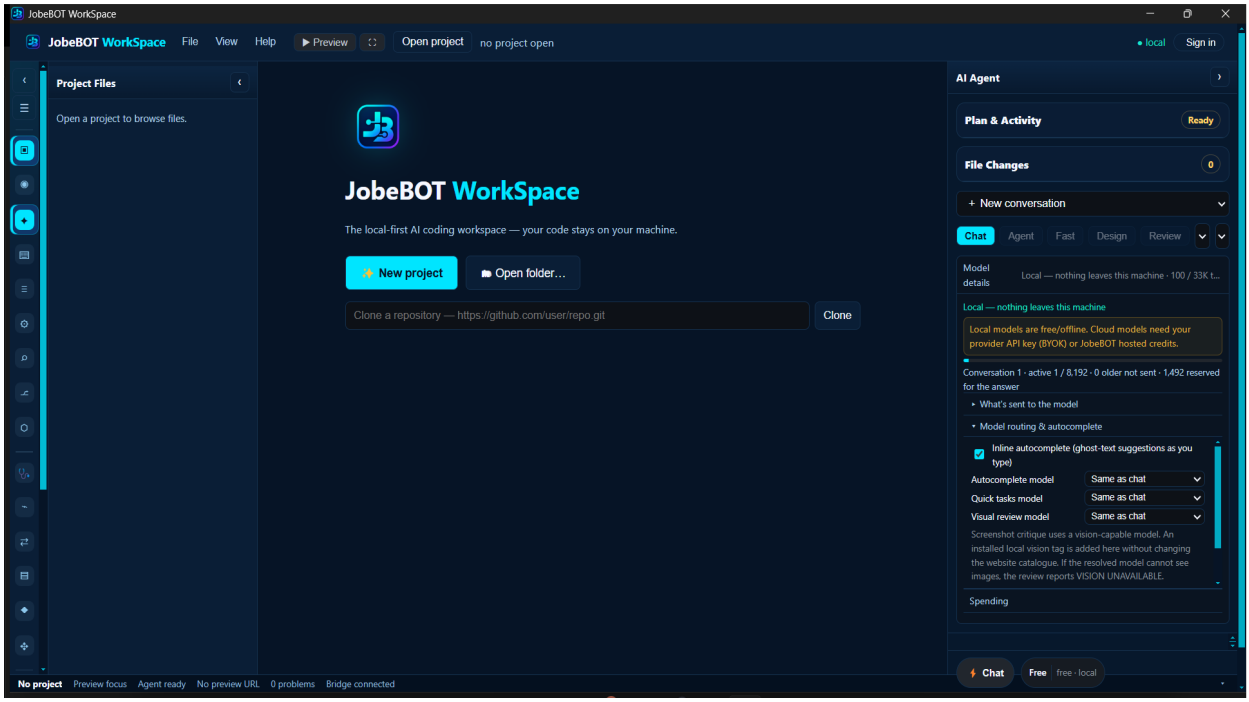
Agents are capability-focused workers. A prompt may use one agent or a coordinated chain. The role determines the kind of judgment expected, while the model supplies inference. A model being listed or responsive does not by itself certify autonomous project work.

Agent role	Best for	Required evidence
Planner or orchestrator	Break down ambiguous work and sequence tasks.	A current plan tied to the actual project.
Coder or file writer	Implement files, tests, and targeted fixes.	Tool calls, read-back, syntax checks, and browser proof.
Reviewer	Inspect diffs, risks, regressions, and tests.	Findings with file and line evidence.

Agent role	Best for	Required evidence
Vision or UI reviewer	Inspect screenshots, layout, overlap, and responsive behavior.	Rendered screenshots or browser evidence.
Security or audit agent	Review permissions, secrets, routes, and release risk.	Concrete findings and reproducible checks.

5. Models and routing

JobeBOT supports local and provider-backed model routes. Local routes include Ollama and LM Studio. Optional gateway routes may include OmniRoute, OpenRouter, NVIDIA NIM, GonkaRouter, Baseten, and other configured OpenAI-compatible services. The catalogue records provider, model identity, capability flags, cost class, privacy notes, and certification state.



Model routing is a policy decision, not only a model picker.

• Discovered means the model is known to the catalogue but not trusted for autonomous work.

• Responsive means a health probe returned, not that tool use or file writing works.

• Certified means the model passed the relevant live capability gate.

• Autonomous routing requires certification, enabled policy, budget authorization, and tool compatibility.

• Free-first routing must not bypass capability or safety gates.

6. Files, tools, and verification

A successful coding run is a chain of evidence. The agent must use project tools, write inside the allowed scope, and verify the result. A response that pastes code in chat is not a disk write. A reported filename is not proof until it is read back from disk.

Tool or check	Purpose
inspect_project	Establish the project shape, files, framework, and current state.
read_file	Read source before editing and verify after editing.
write_file	Create or replace a complete file within approved scope.
run_command	Execute a command only when permitted and relevant.
browser_open or preview	Load the local application and verify the rendered result.
finish	Close the run only when the completion contract is satisfied.

7. Safe operating procedure

• State the project path and desired outcome clearly.

• Name files or user-visible behavior that must change.

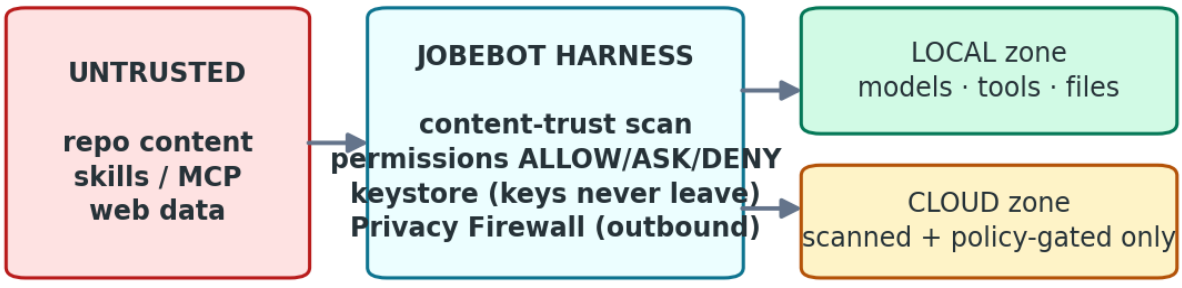
• Ask the agent to inspect before editing when the project is unfamiliar.

• Require a write, read-back, syntax check, and browser check for implementation work.

• Treat provider failures, empty tool calls, stale builds, and missing transcripts as incomplete.

• Review sensitive commands, remote operations, credentials, and production changes before approval.

• Keep backups before replacing existing project files.



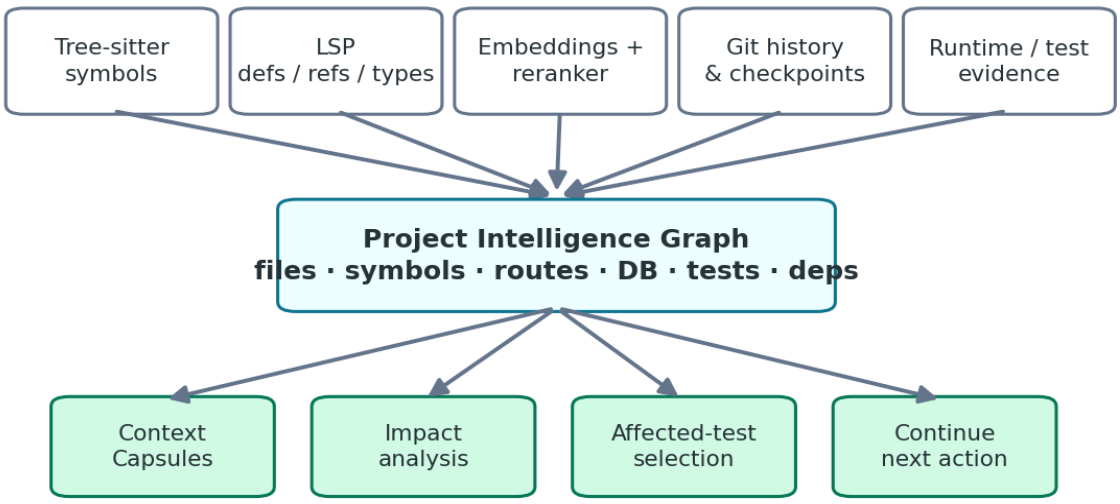
Security boundaries are part of the workflow, not an afterthought.

8. Troubleshooting

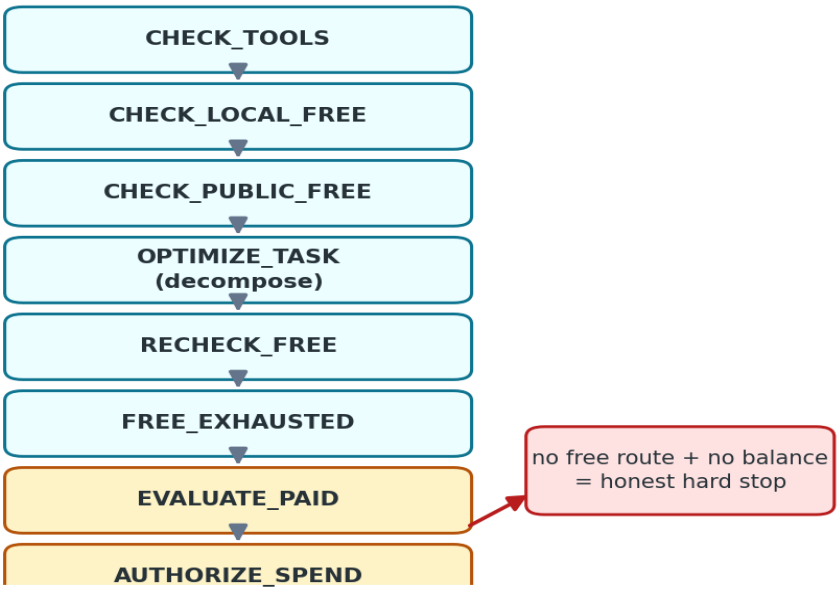
Symptom	Likely cause	First check
The agent only writes prose	Chat mode, missing tool compatibility, or explanation intent.	Confirm Agent mode and run a minimal write probe.
Files are reported but absent	Stale renderer, wrong target, or no read-back evidence.	Compare absolute path, file hash, and read-back content.
The run stays Working	Provider stall, cancellation path, or unresolved approval.	Stop the run and inspect transcript and process state.
Preview is blank or stale	Wrong URL, cache, syntax error, or stopped server.	Check HTTP, assets, console errors, and refresh.
Model is listed but blocked	Discovery is not certification.	Read admission and certification status.

9. Completion checklist

- Requested behavior is visible in the actual target project.
- Intended files changed and unrelated files did not.
- Files exist on disk and match read-back.
- Syntax, tests, HTTP, and browser checks passed where applicable.
- Transcript or audit journal records meaningful steps.
- Final message distinguishes completed, blocked, and unverified work.



Project intelligence keeps the agent grounded in the real repository.



Free-first routing remains capability and certification gated.

This handbook describes the current JobeBOT product architecture and verified operating model. Provider availability, model identities, pricing, and certification states can change; check the live catalogue and current WorkSpace diagnostics before relying on them for a consequential task.