



AI TEAM GUIDE • OCTOBER 2026 • V1.1

AI Models & Inference Deployment, Model Choice & Security

LAST CHANGED

2026-10-13

CONTACT

support@askui.com

Model generations change quickly. This guide reflects the AskUI AI team's recommendations as of October 2026. Confirm with the AskUI AI team or visit docs.askui.com for further information.

TL;DR

The most important take-aways for sales conversations, solution design and customer decisions.

RECOMMENDED

Claude 5.5

Haiku · Sonnet · Opus

DEPLOYMENT OPTIONS

3

AskUI · Own tenancy · On-prem

MODEL MUST HAVE

Vision + Tools

and a long context window

AIR-GAPPED

Supported

License key + local model

-
- 01 AskUI is model-agnostic.** Any model that understands screenshots and supports tool calling can drive the agent. Tests, prompts and reports stay unchanged when you switch models.
-
- 02 We recommend the Claude 5.5 family.** Sonnet 5.5 as the default, Haiku 5.5 or Gemini 3.8 Flash for simple, stable flows, Opus 5.5 for the hardest UIs. If Claude is not an option, the GPT-6 family.
-
- 03 Three deployment options.** AskUI Inference for a zero-config start; your own cloud tenancy (e.g. Azure) via Bring Your Own Model, which most IT-security reviews approve; or fully on-premise and air-gapped with a local model.
-
- 04 Only inference traffic leaves the machine.** Per step: screenshot, step text and test conversation, sent only to the provider you choose. Reports, screenshots, logs and secrets stay local. AskUI does not train models on customer data.
-
- 05 Price per token is not price per result.** Stronger models take more direct paths and fail less, so they can be cheaper per reliable result. The right model can only be determined against the customer's own use case.
-
- 06 Local models work, with limits.** 27–35B open-weight models served with llama.cpp on a DGX Spark or Mac Studio, which AskUI can deliver preconfigured. They handle shorter, simpler tests, especially with well-written tests, rules and UI prompts; use cloud models for the rest.
-
- 07 Measure throughput, not seconds per click.** Agents run in parallel, overnight and on schedules. Some setups reach near-human speed, but daily coverage is what counts.
-

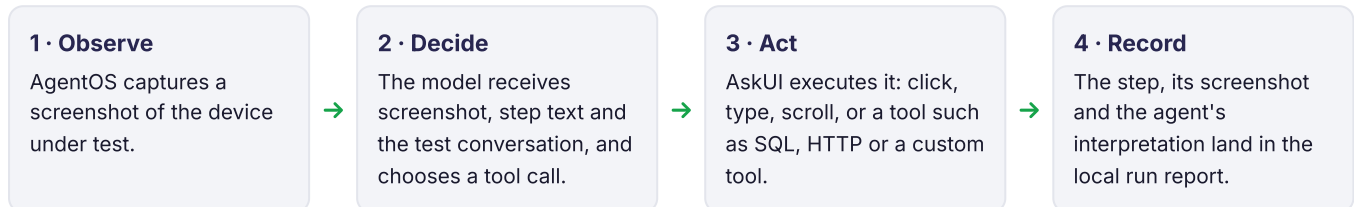
Contents

1	How AskUI uses AI models	4
1.1	The agent loop	4
1.2	The components	4
1.3	What AskUI adds around the model	5
2	Deployment scenarios	6
2.1	Overview	6
2.2	AskUI Inference	6
2.3	Own cloud tenancy (Bring Your Own Model)	6
2.4	On-premise & air-gapped	7
3	Models & how to choose	11
3.1	Recommended models	11
3.2	Requirements	11
3.3	Comparing model tiers	12
3.4	Price per intelligence	12
3.5	Speed in context	13
4	Security	14
4.1	What leaves your network	14
4.2	Securing the inference endpoint	14
4.3	Hardening the machines	15
4.4	Data protection	15
5	Best-practice guide in AskUI Desktop	16
5.1	Connect your own model provider	16
5.2	Run without an AskUI account (air-gapped)	16
5.3	Evaluate models against your use case	16
5.4	Make local or smaller models more reliable	17
5.5	Use the strongest model where it pays off	17
5.6	Keep cost and reach under control	17
6	Frequently asked questions	18

1. How AskUI uses AI models

AskUI agents operate user interfaces the way a human does: they look at the screen, decide on the next action and carry it out. The AI model provides the eyes and the judgement. Everything around it comes from AskUI.

1.1 The agent loop



The loop repeats until the test reaches its goal, fails a check, or cannot continue. A single test usually takes dozens of these steps, and every step is one model call. This is why the model drives both the reliability and the cost of a run.

1.2 The components

AskUI Desktop	Where tests are written, run and reviewed. Starts runs, talks to the model provider, writes reports. Runs on Windows and macOS.
AgentOS	Controls the device under test: screen, keyboard, mouse. Runs on Windows, macOS, Linux, and drives Android and iOS devices. Needs no GPU.
Model provider	The AI service that makes the agent's decisions: AskUI Inference, your own cloud endpoint, or a model server in your network.
AskUI Hub	Sign-in, workspaces, access tokens and usage. Not involved in executing a test, and not needed in license-key mode.
Project	Plain files in your Git repository: tests, plans, prompts, tool configuration, and the reports under <code>agent_workspace/</code> .

WHAT THE MODEL DOES

It reads the screenshot and understands the UI, interprets the test step, plans the next action, chooses which tool to call with which parameters, and judges whether the expected result is on screen. Everything else is AskUI.

WHAT IS SENT TO THE MODEL

Per step, only three things: the current **screenshot**, the **step text** and the **conversation of the current test** (earlier screenshots, decisions and tool results). Testware, reports, logs and secrets never leave your machines. Where the model runs is your choice; see Section 2.

1.3 What AskUI adds around the model

A model alone can operate a screen. Turning that into test automation that stays stable at scale takes the system around it:

Every surface	One agent for desktop and legacy applications, web, Android and iOS, embedded screens and in-car HMIs, and test benches with their peripherals. It tests pixels, not selectors, so it works wherever a human can see the screen.
Tests from requirements	Tests are written in plain language, often derived directly from system or product requirements. No scripting; testers do not need to code.
Stable at scale	Built to stay as reliable at thousands of tests a day as at ten: preconditions, rules files, and four clear results (PASSED, FAILED, BROKEN, SKIPPED), each with its own owner.
Memory	The agent remembers what earlier runs learned about your application: routes, business rules and hazards. It needs no source code and no accessibility tree.
Self-improving suites	Dreaming studies passed and failed runs and proposes small fixes to rules and tests, each as a reviewable diff that a human approves.
Watchdog	In-Flight Guidance reviews a run every few steps and stops runs that loop or drift, with a clean FAIL instead of an endless run.
Forge	Drafts the parts you would otherwise write by hand: UI knowledge from screenshots (PromptForge), custom tools from a description (ToolForge), and tests from recordings and specs.
Audit-ready evidence	Every run produces a structured report: steps, screenshots, the agent's reasoning, expected versus actual result, failure analysis, and a trace to the exact test, device and moment.
Your model, your infrastructure	Model-independent, with your own LLM, in your own cloud, or fully on-premise and air-gapped (Section 2). Secrets, opt-in tools and least-privilege setups keep the agent's reach small (Section 4).

2. Deployment scenarios

The agent, tests and reports are the same in every scenario. What changes is where inference runs, and therefore where screenshots go.

2.1 Overview

Screenshots may be processed by a cloud model via AskUI AskUI Inference Fastest start, no configuration, rate limits handled. Ideal for trials and POCs.	Screenshots must stay within our own cloud contracts Own cloud tenancy (BYOM) Your Anthropic account, Azure AI Foundry, or a corporate LLM gateway. What most IT-security reviews approve.	Nothing may leave our network On-premise / air-gapped License-key mode plus a self-hosted model on local hardware. Enterprise.
--	--	--

SCENARIO	WHERE SCREENSHOTS GO	ASKUI DOMAINS NEEDED	MODELS
AskUI Inference	Through AskUI to the hosted model	auth , workspaces , inference .askui.com	Hosted Claude and Gemini models
Own cloud tenancy	Only to your endpoint, under your contract	auth , workspaces ; none in license-key mode	Claude, GPT, Gemini, or self-hosted in your cloud
On-premise / air-gapped	Nowhere; the model runs in your network	None	Self-hosted open-weight models

2.2 AskUI Inference

The default. Sign in with an AskUI account and run; the agent's model calls are routed through AskUI and billed to your workspace.

- **No configuration** and no provider account needed. AskUI handles rate limits.
- **Hosted models:** the workspace default is a Claude model; other hosted models such as Gemini can be selected.
- **Transparent usage:** every run report shows tokens and cost per test; the AskUI Hub shows the workspace's overall usage.
- **Best for:** trials, POCs, and teams without their own model contract.

2.3 Own cloud tenancy (Bring Your Own Model)

BYOM points the agent at a model endpoint you control. It changes only *where inference runs*; agent, prompts, tools and reports are identical. You pay your provider directly, and AskUI charges no inference fees.

API TYPE	WORKS WITH	AUTHENTICATION
Anthropic Messages API	Anthropic, Azure AI Foundry, Anthropic-compatible gateways and servers. The tested path.	API key
OpenAI-compatible (Chat Completions)	OpenAI, OpenRouter, corporate LLM gateways, self-hosted servers. Best-effort support.	API key, none (local servers), or Microsoft Entra ID

EXAMPLE · MICROSOFT AZURE

In Azure, your isolated network is a **Virtual Network (VNet)**; AWS and GCP call it a VPC. Two patterns keep inference inside it:

- **Managed model:** deploy a Claude or GPT model in an EU region of your Azure AI Foundry resource and connect AskUI to it. Make it reachable only from your VNet (Section 4.2).
- **Self-hosted model in the VNet:** run an open-weight model with llama.cpp on a GPU VM inside your VNet and connect via its private address. No model provider outside your tenant is involved; the limits of Section 2.4 apply.

AWS Bedrock and GCP Vertex AI use their own authentication schemes. Put a gateway in front that exposes an Anthropic- or OpenAI-compatible endpoint, or talk to us. Strict corporate gateways can be accommodated with a small message transform in the project.

2.4 On-premise & air-gapped

Self-hosting keeps every screenshot inside your network and removes per-token cost. In exchange, today's open-weight models are clearly weaker than frontier cloud models on complex tests. Choose it when data residency matters more than peak robustness.

SETUP AT A GLANCE

1. **License-key mode** instead of an AskUI account: no sign-in, no AskUI domains, no AskUI Inference.
2. **A model server in your network** running a 27–35B open-weight vision model with llama.cpp.
3. **Internal mirrors** for runtime downloads (browser bundles, Android SDK, NuGet packages for custom tools), or pre-provisioned machines.
4. **Offline distribution** of AskUI installers and updates.

Model size classes

Self-hosted models must have vision and tool calling and a context window of at least 128k (ideally 200k). We only consider open-weight models released in 2026.

CLASS	EXAMPLES (OPEN WEIGHTS)	STATUS	FIT
~7–9B	Small variants of current model families	NOT RECOMMENDED	No good results so far: too weak to plan and verify test steps reliably.
~30B	Qwen 3.8 27B (dense) Qwen 3.6 35B (3B active) Meta Muse Glimmer 30B	LIMITED	Shorter, clearly guided tests. Runs on a single DGX Spark or Mac Studio.
~70–120B	Mistral Small 4 119B (6.5B active)	EVALUATION PENDING	Expected to close part of the gap to cloud models. Native vision and tool calling.
~400B	GLM-5.3-Flash 320B (18B active)	EVALUATION PENDING	Natively multimodal, 1M context. Still practical on-premise with large unified memory or a small GPU server.
>1T	Kimi K3 2.8T (~50B active) Mistral Large 4 1.05T (~52B active)*	EVALUATION PENDING	Frontier-scale open weights, natively multimodal. Requires data-centre hardware.

Most models above ~70B are mixture-of-experts (MoE): only the active parameters are computed per token, which keeps them fast, but **all parameters must be held in memory**. Size memory by total parameters. Candidates in the larger classes have not yet been evaluated by AskUI; check with the AI team before committing to one. *Mistral Large 4: in public preview since October 2026; open weights announced for the end of October 2026.

How much memory do you need?

RULE OF THUMB

$$\text{Memory} \approx (\text{weights} + \text{KV cache}) \times 1.2$$

- Weights (GB)** $\approx$ total parameters in billions $\times$ bytes per parameter: 2 at 16-bit, 1 at 8-bit, about 0.55 at 4-bit quantization.
- KV cache (GB)** $\approx 2 \times$ layers $\times$ KV heads $\times$ head size $\times$ bytes $\times$ context tokens $\times$ concurrent runs $\div 10^9$. Take the first four values from the model's configuration.
- $\times 1.2$** covers activations, the vision encoder and runtime overhead.

Example: a model with 64 layers, 8 KV heads and head size 128 at 16-bit holds about 0.26 MB per token, so **~52 GB for one run at 200k context**. Newer models with hybrid or linear attention need a fraction of that; 8-bit KV-cache quantization halves it.

CLASS	WEIGHTS AT 4-BIT	WEIGHTS AT 8-BIT	PLAN FOR (ONE RUN, 200K CONTEXT)	TYPICAL HARDWARE
~30B	~15–20 GB	~30–35 GB	48–96 GB	One DGX Spark or Mac Studio
~70–120B	~40–65 GB	~70–120 GB	96–192 GB	DGX Spark (4-bit), two clustered DGX Sparks, or a large Mac Studio
~400B	~180–220 GB	~320–400 GB	250–550 GB	Mac Studio with maximum memory, a DGX Spark cluster, or a multi-GPU server
>1T (e.g. 2.8T)	~1.5 TB	~2.8 TB	2–3.5 TB	Multi-node GPU cluster in a data centre

DISCLAIMER

These are planning estimates, not guarantees. Three factors change the result decisively: the **number of concurrent runs** (each needs its own KV cache), the **context length and KV-cache settings** (caching, cache quantization), and the **quantization** of the weights (lower precision saves memory but can cost accuracy). Size the server for your expected parallel load, and validate with your own tests before buying hardware.

Serving stack

USE	STACK	NOTES
Production	llama.cpp	Recommended. Enable KV caching and flash attention, prevent model offloading, keep the build current (new architectures need recent builds).
Experimentation	LM Studio, Ollama	Quick to try on a workstation. Check the configured context length; defaults are often far too small.
Connection	OpenAI-compatible API	AskUI connects to the server's <code>/v1</code> endpoint (Section 5.1).

Hardware

OPTION	STATUS	WHY
NVIDIA DGX Spark	RECOMMENDED	128 GB unified memory and strong compute for processing screenshots. Can be clustered. AskUI can deliver a preconfigured setup.
Apple Mac Studio (M5 Ultra)	RECOMMENDED	Very large unified memory and a strong chip. AskUI can deliver a preconfigured setup.
Workstation with GPU(s)	LIMITED	Usually too little VRAM: even ~30B models need 48 GB or more with a long context, larger classes far more.
Laptop / consumer machine	NOT RECOMMENDED	Only fits small models, which do not work reliably.

Test machines need no GPU. Inference runs on the model server; test machines only host the application under test.

Examples and counter-examples

WORKS	DOES NOT WORK
DGX Spark or Mac Studio running llama.cpp with a 27–35B vision model and a 200k context	A 7B model in Ollama on a tester's laptop
LM Studio on a strong workstation to check whether a UI is feasible	LM Studio or Ollama as the production inference server
Short, clearly guided HMI tests on a local model, with precise rules and UI prompts	Long, branching end-to-end flows across several applications on a local model
A current llama.cpp build that supports the model's architecture	An outdated runtime that does not support the model yet

3. Models & how to choose

3.1 Recommended models

PROVIDER	MODELS	AVAILABILITY	STATUS
Anthropic Claude	Haiku 5.5 Sonnet 5.5 Opus 5.5	AskUI Inference, your own Anthropic key, Azure AI Foundry, or an Anthropic-compatible gateway	RECOMMENDED Primary, tested path
OpenAI GPT	GPT-6 Luna GPT-6.1 Sol GPT-6 Astra	Own tenancy via an OpenAI-compatible API: OpenAI, OpenRouter, or your gateway	SUPPORTED Alternative if Claude is not available
Google Gemini	Gemini 3.8 Flash	AskUI Inference, or own tenancy via an OpenAI-compatible gateway	SUPPORTED Fast and cost-efficient
Qwen (open weights)	Qwen 3.8, 27B–35B class	Self-hosted	LIMITED Simpler tests
Meta Muse Glimmer	30B	Self-hosted	LIMITED Shorter, simpler tests
Larger open weights	~70–120B, ~400B and >1T classes	Self-hosted (Section 2.4)	EVALUATION PENDING

3.2 Requirements

A MODEL MUST HAVE

- **Vision:** the agent drives the UI from screenshots.
- **Tool / function calling:** every action is a tool call.
- **Context window:** 200k tokens recommended; 128k works for simpler tests but raises the risk of a run aborting.
- **Max output tokens:** a safety limit, not a tuning knob; 32k is always safe.

WILL NOT WORK

- **Text-only models:** they connect, but cannot see the screen.
- **Models without tool calling:** they cannot execute a step.
- **Pure grounding or pointing models:** they locate elements but cannot plan a test.
- **Small models (7–9B):** no good results so far.

3.3 Comparing model tiers

MODEL TIER	SPEED ↑	PRICE PER TOKEN ↓	ROBUSTNESS ON COMPLEX UIS ↑	TYPICAL USE
Claude Haiku 5.5 GPT-6 Luna Gemini 3.8 Flash	●●●	●●●	●●●	Simple, clearly guided flows; smoke tests; high-volume regression on stable UIS
Claude Sonnet 5.5 GPT-6.1 Sol	●●●	●●●	●●●	Default. Best balance for most test cases
Claude Opus 5.5 GPT-6 Astra	●●●	●●●	●●●	Hardest UIS, long or ambiguous flows, unusual or embedded HMLs
Self-hosted ~30B	Depends on hardware	No per-token cost; hardware investment grows with size	●●●	Data must not leave the network; shorter, simpler tests
Self-hosted ~70–120B			PENDING	Evaluation pending
Self-hosted ~400B			PENDING	Evaluation pending
Self-hosted >1T			PENDING	Evaluation pending

↑ higher is better · ↓ lower is better · more dots are always better. Qualitative ranking by the AskUI AI team, not benchmark results. Models in one row are comparable tiers; Claude is the path AskUI tests against.

3.4 Price per intelligence

The price per token is not the price per test. What a customer pays for is a **reliable test result**, and that depends on how many steps the agent needs and how often a run has to be repeated:

Cost per reliable result = (steps × tokens per step × price per token) ÷ share of runs that succeed

plus the human time spent investigating failed or unclear runs

- **Stronger models take more direct paths.** They read the UI correctly the first time, need fewer steps, and get stuck less often.
- **Weaker models can cost more overall.** Detours, retry loops and failed runs that must be re-run or investigated quickly eat up a lower token price.
- **So Opus 5.5 can be cheaper than Haiku 5.5** on a demanding use case, while Haiku is the better deal on simple, stable flows.
- **There is no general answer.** Which model gives the best price per intelligence can only be determined against the customer's own use case. Section 5.3 shows how to evaluate it in AskUI Desktop.

3.5 Speed in context

- Each step takes a few seconds, depending on model and hardware. With the right setup, AskUI reaches **near-human execution speed**.
- **Comparing a single run with a human tester is the wrong measure.** Agents run in parallel across many machines, overnight and on schedules, without breaks, and produce the same evidence every time. What matters is throughput and coverage per day, not seconds per click.
- Faster models are not automatically faster tests: a model that needs fewer steps and no re-runs often finishes first.

4. Security

4.1 What leaves your network

Only the inference traffic: screenshot, step text and the conversation of the current test, sent to the model provider you chose. Everything else stays local:

Testware	Project folder in your repository, plain files
Reports, screenshots, logs	<code>agent_workspace/</code> inside the project, local. Nothing uploads them.
Credentials	Named secrets, encrypted with your OS user account, hidden from the model, redacted from reports and logs
Model API keys	Encrypted on the device; never written to reports or logs
AskUI Hub	Sees sign-in, tokens and usage counters, not test content. Not used in license-key mode.
Telemetry	AskUI collects usage data to improve the product experience. Telemetry can be disabled.

4.2 Securing the inference endpoint

OWN CLOUD TENANCY, E.G. AZURE

- Deploy in an **EU region**.
- Use a **private endpoint** and disable public network access, so the model is reachable only from your VNet.
- Prefer **Microsoft Entra ID** over static API keys where your gateway supports it.
- Check the data-processing terms of the specific model in your cloud contract.
- Route through your corporate proxy (system proxy, NTLM/Kerberos supported).

ON-PREMISE / AIR-GAPPED

- **License-key mode:** no AskUI domains at all.
- Model server on an internal network segment, reachable only from the machines that start runs.
- Internal mirrors for browser bundles, Android SDK and NuGet; offline installers.
- In-Flight Guidance with a local reviewer model, if used.

4.3 Hardening the machines

The agent can do what the signed-in user can do, through the tools it has been given. Keep both minimal:

- **Dedicated local test account:** standard user, no administrator or domain-wide rights, nothing personal in the profile.
- **Elevation off** unless a suite needs it; give such a suite its own machine.
- **Test users in the application** with only the role the scenario needs. Never production or personal accounts.
- **Tools are opt-in.** Shell, SQL, HTTP and file tools are off until enabled, and every grant is visible in the project for review.
- **Custom tools and MCP servers** are code and third-party reach; review them like production code.
- **Network:** test machines reach only the system under test; the inbound AgentOS port is open only to machines allowed to start runs.

4.4 Data protection

- Screenshots can contain personal data if your test data does. Prefer synthetic or anonymised test data, and choose the inference placement accordingly.
- Reports and evidence are files in your project, so your own retention and deletion rules apply.
- AskUI is ISO 27001 certified and GDPR compliant, and does not train models on customer data. With your own model provider, data handling follows your contract with that provider.

5. Best-practice guide in AskUI Desktop

Practical recommendations from the AskUI AI team, with the exact place to find each setting in AskUI Desktop.

5.1 Connect your own model provider

SETTINGS → ACCOUNT → MODEL PROVIDER

1. Behind a corporate proxy, first configure **Settings → Network proxy**.
2. Open **Settings → Account → Model Provider** and pick the API type: **Anthropic** for Claude (Anthropic, Azure AI Foundry), **OpenAI-compatible** for everything else, including your local model server.
3. Enter base URL, model and key. For a local server, the base URL ends in `/v1` and the key can stay empty. For an Entra-protected gateway, choose **Microsoft Entra** and enter tenant, client ID, scope and secret.
4. Click **Save**. AskUI makes a test call first, so a wrong key, URL or model shows up here and not mid-run.

To return to AskUI Inference, choose **Remove key** in the same place.

5.2 Run without an AskUI account (air-gapped)

WELCOME SCREEN → ACTIVATE WITH A LICENSE KEY

1. On the welcome screen, choose **Activate with a license key** instead of signing in.
2. Paste the license key and continue.
3. Add your model provider (as in 5.1). License-key mode always needs one, since there is no AskUI Inference.

License and plan tier are shown in the profile menu (top right). Removing the key there frees the seat for another machine.

5.3 Evaluate models against your use case

RUN REPORTS · AskUI HUB → USAGE

1. Pick **3–5 representative tests**: a simple one, a typical one, and the hardest flow you need.
2. Start with **Sonnet 5.5**, then run the same tests on a cheaper model (Haiku 5.5, Gemini 3.8 Flash) and, for the hard cases, on Opus 5.5. Run each test **several times per model**; a single run says little.
3. Compare in the run reports: status, number of steps, duration and cost per test.
4. Decide by **cost per passed run** (total cost ÷ passed runs), not by price per token (Section 3.4). Models can differ per test plan.

Write tests in English and with precise steps; models reason best in English, and clear steps save steps and money with every model.

5.4 Make local or smaller models more reliable

[TESTS](#) · [RULES](#) · [UI PROMPTS](#)

First and most important: give the model better instructions. Smaller models need more guidance than frontier models, and that guidance lives in your project:

1. **Tests:** precise, step-by-step steps with an explicit expected result per step and clear preconditions and postconditions.
2. **Rules** (`tests/rules.md`): project-wide behaviour, e.g. how to handle dialogs, logins or waiting times.
3. **UI prompts** (`tests/ui.md`): what the agent should know about your UI, e.g. icon meanings, hidden controls or domain terms. PromptForge can draft them from screenshots.
4. Read the run reports of failed runs: the agent's reasoning shows exactly where it misread the UI, and therefore what to add.

Additional instrument, not the default: In-Flight Guidance (**Extending** → **In-Flight Guidance**, experimental) lets a second, ideally stronger reviewer model check the run every N steps and nudge it back on course or stop it with a clean FAIL. Use it on top of well-written tests, not instead of them.

5.5 Use the strongest model where it pays off

[SETTINGS](#) → [ACCOUNT](#) → [FORGE MODEL](#)

Forge drafts tests, UI prompts and custom tools for you. It runs only occasionally, so the strongest available model is worth it, even if your test runs use a cheaper one. On your own provider, untick **Same as for my test runs** to configure a separate one.

5.6 Keep cost and reach under control

[EXTENDING](#) → [SECRETS](#) · [TOOL STORE](#)

- **Short, focused tests:** one goal per test. Watch step counts in the report; a step retried many times multiplies cost.
- **Usage:** tokens and cost per test in the run report; overall workspace usage in **AskUI Hub** → **Workspace** → **Usage**.
- **Credentials** only as named secrets in **Extending** → **Secrets**, never in test files.
- **Tools** from **Extending** → **Tool Store** only when a test needs them, and scoped: fixed scripts, SQL allowlists, allowed hosts.

6. Frequently asked questions

Collected from real conversations with customers and prospects in 2026, ordered by how often they come up.

DEPLOYMENT & OWN MODELS

Can we use our own LLM, API key or endpoint, e.g. Azure AI Foundry, Bedrock, Vertex or an internal GPT gateway?

Yes. AskUI connects to any Anthropic-compatible or OpenAI-compatible endpoint, including gateways protected by Microsoft Entra ID. Bedrock and Vertex need a compatible gateway in front, or a short alignment with us. See Sections 2.3 and 5.1.

Can AskUI run fully on-premise or air-gapped, with no data leaving our network?

Yes. In license-key mode with a self-hosted model, no data goes to AskUI or any other third party. See Sections 2.4, 4.2 and 5.2.

What hardware do we need for a local model, and can one machine serve several users?

For the ~30B class, one DGX Spark or Mac Studio, which AskUI can deliver preconfigured; larger classes need more memory. One model server can serve several runs, but each concurrent run needs its own KV cache. Section 2.4 has a sizing rule of thumb and a table per model class.

Which open-weight models can we use, and which settings?

Vision-capable models with tool calling, released in 2026, from the ~30B class upwards, served with llama.cpp. Configure a context window of 200k (at least 128k) and 32k max output tokens. See Sections 2.4 and 3.2.

How much do we lose with a local model, and how much can run locally?

Local ~30B models handle shorter, clearly guided tests; complex flows need a frontier model. A hybrid setup works well: simple tests locally, hard ones on a cloud model. See Sections 3.3 and 5.4.

DATA & SECURITY

What data is sent where? Are screenshots stored?

Per step, only the screenshot, the step text and the test conversation go to the model provider you chose. Reports, screenshots and logs are stored only in your project. See Sections 1.2 and 4.1.

Is it GDPR-compliant and processed in the EU?

Yes. AskUI is GDPR compliant and AskUI Inference is processed in the EU. With your own provider, you choose the region yourself. See Sections 4.2 and 4.4.

Do AskUI or the model providers train on our data?

AskUI does not train models on customer data, and AskUI Inference uses provider agreements that exclude training. With your own provider, your contract with that provider applies. See Section 4.4.

Are credentials sent to the model?

No. Tests reference secrets by name; values are inserted locally at execution time and redacted from reports and logs. See Sections 4.1 and 5.6.

Do you have security documentation for our IT or AI governance review?

Yes: data-flow diagram, ISO 27001 certificate and penetration-test summary are available in our Trust Center under NDA. Section 4 summarises the essentials for a first review.

COST

What will inference cost per test or per month at our volume?

It depends on the model, the test length and how reliably the model reads your UI; a few complex tests usually drive most of the spend. The reliable way to estimate it is a short evaluation on your own tests. See Sections 3.4 and 5.3.

Is inference included in the licence? How do we track spend with our own model?

No. With AskUI Inference, usage is metered against your workspace. With your own provider, AskUI charges no inference fees; run reports show tokens per test, and you track spend in your provider's dashboard. See Sections 2.2, 2.3 and 5.6.

How do we avoid paying for the model on every regression run?

Choose the cheapest model that passes reliably per suite, keep tests short and precise, and consider self-hosted models, which remove per-token cost entirely. See Sections 3.4, 2.4 and 5.6.

MODEL CHOICE, SPEED & RELIABILITY

Which model do you recommend?

Claude Sonnet 5.5 as the default, Haiku 5.5 or Gemini 3.8 Flash for simple flows, Opus 5.5 for the hardest UIs; comparable GPT-6 tiers if Claude is not an option. Decide by cost per reliable result. See Sections 3.1, 3.3 and 3.4.

Why is an agentic test slower than a script?

Every step is a model decision on a screenshot, which takes a few seconds; some setups reach near-human speed. More importantly, agents run in parallel and around the clock, so throughput per day is the better measure. See Section 3.5.

Our runs are slow. Why, and what can we do?

Check these causes in this order:

- **Many steps per test:** look at the step count in the run report. Detours and retry loops usually mean the agent misreads the UI; sharpen the test steps, rules and UI prompts (Section 5.4).
- **Long tests:** the conversation grows with every step, so later steps take longer. Split long tests into short, focused ones (Section 5.6).
- **Large screenshots:** 4K screens make every step heavier; Full HD is usually enough.
- **Model choice:** try a faster tier on simple flows, but compare total run time, not time per step; a stronger model often finishes first because it needs fewer steps (Sections 3.3 and 3.4).
- **Self-hosted model server:** too little memory or compute, or too many concurrent runs on one server, slows every step. Check the sizing in Section 2.4.

Is it deterministic? What about false positives?

Model output varies between runs. Precise, step-by-step tests with explicit expected results make behaviour consistent, and every verdict is backed by screenshots and the agent's reasoning in the report. See Sections 1.3 and 5.4.

How does the agent pick the right control among similar ones? Can we give it domain knowledge?

Yes. Describe your UI's specifics in the project's UI prompts and rules, for example icon meanings or domain terms; PromptForge can draft these from screenshots. See Section 5.4.

FURTHER READING · [DOCS.ASKUI.COM](https://docs.askui.com)

- Model providers (BYOM): docs.askui.com/docs/extending/model-providers
- Enterprise fit: docs.askui.com/docs/concepts/enterprise-fit
- What the agent may do: docs.askui.com/docs/best-practices/security
- Network requirements: docs.askui.com/docs/reference/network-requirements
- In-Flight Guidance: docs.askui.com/docs/extending/in-flight-guidance

Questions: AskUI AI Team · sales@askui.com · support@askui.com