GREMLIN*

PRODUCT LITEPAPER

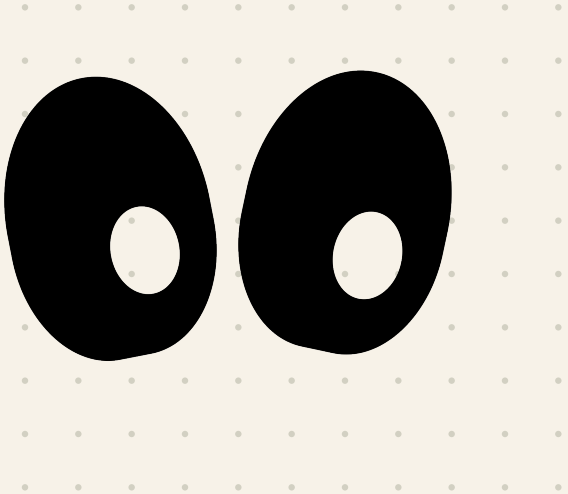
NOVEMBER 2026

Gremlin litepaper

Uncensored models and token powered inference

Gremlin is building a user-directed AI workspace with uncensored model choices, free local inference, and paid access to large hosted models. The Gremlin ecosystem token is intended to connect subscriptions and inference consumption to useful access within the product.

This litepaper explains the product thesis, the role of the token, and the current implementation. It separates existing software from proposed commercial features and does not set token terms or launch pricing.



VERSION 0.1

getgremlin.xyz

01

Why we want uncensored models

People should be able to explore difficult ideas, write freely, and decide how an AI assistant helps them.

More room to think

We want models that engage with legitimate questions instead of stopping at a broad refusal. Creative work, controversial research, debate, and technical exploration often need context and nuance. Unnecessary refusals interrupt that work and leave the user with less control over their own tools.

Choice belongs with the user

Different people want different model behavior. Gremlin brings reduced-refusal models into a workspace where users choose their model, instructions, and permitted tools. Local options also give users a way to run inference on their own hardware rather than depending entirely on one hosted provider.

What uncensored means here

We use uncensored to describe models tuned or modified to reduce refusal behavior. Abliteration is one family of techniques that alters model internals to reduce learned refusals. These approaches can change capabilities as well as behavior; they do not guarantee that every request will be answered or that answers will be correct.

Evaluate the work

A useful model must still follow instructions, reason well, and handle tools reliably. We care about those qualities alongside openness. Reduced refusal is a model characteristic, not a benchmark result, a guarantee of neutrality, or permission to ignore laws and provider rules.

Our aim is greater user agency with honest information about what each model can and cannot do.

02

Local freedom and hosted scale

Use your device when it is enough. Use the hosted service when the job needs more capacity.

Local models

Supported models run in the browser using the device GPU. Local inference has no Gremlin inference charge and remains available after a trial or pass expires. Users still supply the hardware, storage, power, and initial download. Model size and browser support determine what a device can run well.

Large hosted models

Large models can exceed the memory and compute available on everyday devices. Gremlin's hosted route makes selected models accessible without a local download or a personal provider key. The current premium catalog names Hermes 4 405B and Venice Uncensored / Dolphin Mistral 24B. Availability depends on activation and compatibility testing.

Bring your own provider

Users can also connect supported external model providers with their own credentials and credits. Those provider costs are separate from Gremlin-funded inference. This route gives experienced users another choice of model and billing relationship.

One workspace

Gremlin combines chat, missions, notes, tasks, user-editable memory, skills, and approved tool connections. Wallet sign-in ties hosted records to an account. Wallet ownership does not mean every record is stored on-chain or every computation is private. Local inference, synced storage, and external tools have distinct data boundaries.

The current hosted path is provider-backed through OpenRouter. This litepaper does not claim that Gremlin already operates its own GPU fleet.

03

The token powers useful access

The Gremlin ecosystem token is intended to power access to the large hosted models that users choose to run.

Subscriptions

Users buy time-based access to the hosted service. A subscription can include a clearly stated inference allowance for its service period. The current implementation has trial and paid-pass entitlements and checks them on the server before funded requests can run.

Metered inference

Users pay for the inference they consume through the service's published allowance and usage rules. Model choice and request size affect the underlying compute cost. A subscription is not a promise of unlimited generation. Separate usage purchases or top-ups are a proposed commercial extension, not a verified feature of the current build.

The current token mechanism

The source includes a token-burn path for renewing a paid pass. It requires a configured burnable ERC-20 contract and verified transaction evidence that supply was actually reduced. The exact chain, contract, amount, and service terms must be published and tested before customers rely on this payment route.

A sustainable service

Hosted inference creates real operating costs: provider charges or GPU capacity, networking, storage, and service operations. Burning a token does not transfer money to an infrastructure provider. The operator must separately fund compute and maintain a service budget that supports the access it sells.

The utility described here is access and consumption within Gremlin. No supply schedule, allocation, staking reward, buyback, revenue share, price appreciation, or investment return is promised.

04

How hosted inference works

A paid service needs clear limits before a request and reliable accounting after it.

1	**Choose**	The user selects an available model and submits a request.
2	**Authorize**	The server verifies the wallet session, entitlement, and allowed model.
3	**Reserve**	The service reserves capacity in the user's allowance and the shared budget.
4	**Run**	The selected provider performs inference; Gremlin returns the response.
5	**Reconcile**	Confirmed usage updates the allowance. Uncertain charges stay reserved.

Compute is a finite resource

Large-model requests consume accelerator time and memory. Long inputs, long outputs, repeated tool steps, and paid research can increase cost. Published limits should show the available allowance, its reset date, and any separate charges before a user commits.

Current service boundaries

The funded route currently covers selected text models only. Images and video are outside that allowance. Requests must fit both the per-wallet allowance and the service-wide budget. Provider availability, model compatibility, and configured spending limits can stop new requests even during an active pass.

Pricing remains a published product choice

This paper fixes no subscription price, exchange rate, token quantity, or per-request tariff. Current technical defaults are configurable allowances, not launch pricing. Any future credit purchase or overage feature needs clear terms, receipts, and spend controls before release.

05

User agency with clear boundaries

A model can be more willing to answer while the user stays in control of what the software does.

Approve consequential actions

The agent harness validates tool arguments and asks for the applicable approval before actions such as writing files, operating a shared browser tab, or submitting paid generation. A model response cannot grant itself new permissions. Stopping a run prevents later actions but cannot undo an action already dispatched.

Choose what leaves the device

Local generation runs on the device, while synced workspace records can still be stored by Gremlin. Hosted inference sends selected context to a provider, and connected tools can send data to their services. Saved memory has separate cloud-sharing controls. Users should review the actual route and provider policies before submitting confidential material.

Keep an editable memory

Users can edit the agent's working style and remembered facts. The current implementation supports device storage, export, and optional manual sync. Browser storage can be cleared or evicted, and it is not a guarantee of encryption. External copies and provider records have their own retention and deletion rules.

Make limits visible

Models can hallucinate, miss context, and make poor decisions. Reduced-refusal modifications can also affect performance. Gremlin's structured-output and tool checks help catch protocol failures; they are not proof of correctness. Users should verify important outputs and remain responsible for lawful use.

Gremlin's current agent loop runs while the app is open. Saved checkpoints and inbound bot messages do not establish always-on autonomous execution.

06

Current build and next steps

The foundation exists in code. Premium availability and commercial readiness still require verification.

Implemented foundation

The repository includes wallet authentication and owned workspaces; local and bring-your-own-key text routes; hosted entitlements and budget reservations; supervised tools; skills; memory controls; and a token-burn pass verifier. Feature availability on a live deployment depends on configuration and testing.

Before premium activation

Verify the actual token contract and renewal flow. Complete live compatibility checks on each advertised premium model and research route. Confirm provider funding, accounting, data disclosures, and real-device behavior. The source documents a successful temporary test-model run, not certification of the Hermes and Venice premium paths.

Proposed commercial work

Publish subscription terms, allowance behavior, renewal conditions, and clear usage pricing. Design and validate any additional inference-credit or top-up flow. These are proposed next steps without a committed release date. Always-on execution and dedicated GPU hosting are not assumed to exist.

SOURCE NOTES

01 Product and current implementation

02 Harness and inference accounting

03 Deployment and activation requirements

04 Hosted model catalog

Prepared 8 November 2026 from repository commit 8be135d. This is a product litepaper, not an offer of securities or a promise of financial returns. See current product disclosures before using a paid service.