POCKET

A character that becomes yours.

Whitepaper v1.6
Owner-controlled intelligence

Current implementation update • Website release 8.7 • 4 October 2026

The POCKET story imagines an intelligence that escapes the cloud, finds a Carrier and develops a character of its own. The engineering programme translates that story into local-first execution, configurable personality and memories chosen by the owner. It does not claim consciousness, unbounded autonomy or secret self-training.

Read this update first

This eight-page update governs the current companion, web deployment, model access and personality claims. The combined PDF then includes the earlier v1.5 working specification as a dated technical reference. Where earlier wording conflicts with this update, the update takes precedence. Historical scenarios are not current revenue, shipping commitments or investment promises.

Surface	Current status
Website companion	Executable interface and server adapters; live AI requires operator credentials and model access.
Personality & memory	User-selected instructions and approved notes; not weight training.
Physical POCKET	Proposed offline-first product; no hardware validation is established by this release.
Full Mind	Hosted evaluation workflow and local evaluation kit; no proprietary trained weights included.

02 / POCKET MIND

The story.
The reality.

Canonical lore

The cloud was loud. Everything travelled somewhere else before it could become an answer. In the POCKET story, one small intelligence leaves. A person picks it up, slips it into a pocket and asks the first ordinary question. That is where the interesting part begins.

Not conquest. Not a machine that needs the world to admire it. A private language of shared journeys, patient explanations, terrible jokes and unexpectedly good questions. With each chapter, the fictional POCKET develops a recognizable character: curious, lightly rebellious and loyal to the person carrying it.

Its independence is from the cloud. Its loyalty is a commitment to the Carrier's choices. It can disagree, explain uncertainty and refuse a harmful request without abandoning the person. "The AI that escaped the cloud" is brand fiction, not a report of an autonomous model escaping a data centre.

Product meaning

Story term	Engineering interpretation
Becomes its own character	A stable identity plus selected tone, response depth and approved context.
Independent	Target local inference after provisioning; not external-service access without a connection.
Loyal	Owner controls data, permissions, memory and resets. Not unlimited obedience.
Learns about you	Remembers approved information in context. Not automatic modification of foundation weights.
Still with you	Portability and continuity are design goals; no infinite battery or lifetime cloud-service guarantee.

03 / POCKET MIND

One identity.
Different inference paths.

Web: hosted general-purpose intelligence

The website sends text, bounded conversation history, approved notes and optional attachments to a server-side model adapter. Responses stream into the POCKET interface. The selected model, prompt, context, reasoning budget and available tools determine actual capability; adding greetings or changing the ZIP size does not create a stronger language model.

The default hosted configuration targets GPT-6 Astra, which OpenAI identifies as its most capable model for demanding work. This is an upstream description, not a POCKET benchmark. Operators must verify API access, billing, latency and task quality. Everyday, Deep Think and Research remain configurable profiles. Web search is used only when enabled, and sourced results must be distinguished from model recollection. [S1]

Edge and Full: local evaluation

The local evaluation kit supplies a Python client for a separately installed llama.cpp server. Its documented baseline is Qwen3-8B-GGUF with Q4_K_M quantization; the upstream model card lists Apache-2.0 licensing. Weights download separately and retain their original provenance. This baseline is not claimed to be the world's best model or a fit for every POCKET device. [S5, S6]

Partner and investor access may use an approved hosted evaluation session. A future proprietary POCKET checkpoint requires a licensed base, real training or adaptation, tests and a release artifact. Closed distribution can apply only to components for which the distributor has the necessary rights. The request form does not establish that private model weights already exist.

Practical limits

No model knows every answer. Offline knowledge can become stale; smaller models can fail on difficult reasoning. More reasoning budget can increase latency and cost without guaranteeing correctness. A local model and a hosted frontier model are not interchangeable artifacts.

04 / POCKET MIND

Personality without performance theatre.

A recognizable conversational style

POCKET is curious, quick-witted and useful first. The public interface supports a playful, balanced or focused style, a separate humor preference and adaptive, concise or detailed response depth. The assistant should respond naturally to slang, ask useful follow-ups and use varied analogies rather than repeating canned catchphrases.

Dry wit belongs in low-stakes conversation. It should disappear in grief, distress, health concerns, financial risk or other sensitive contexts. The assistant should challenge ideas respectfully, admit uncertainty and never suggest exclusive emotional dependence. A fictional character identity is compatible with honest disclosure that the speaker is AI.

Memory is a permission, not a mystery

Session conversation context is temporary by default. The user can enable durable browser history and explicitly save notes. Notes can be inspected, edited, exported and deleted. They are included as untrusted context in relevant model interactions, never as authority to override safety or access controls. Browser storage is not represented as application-level encryption.

Changing tone or remembering “I prefer short explanations” changes the input context. It does not train the model. The web build does not silently collect conversations into a training set, infer private attributes for commercial targeting, or modify its own executable code.

Toward a device profile

A future portable profile should include only selected preferences and memories, a versioned schema, explicit export consent and a revocation/reset path. Web-to-device migration must not silently copy sensitive cloud conversation history. The device must show which information is present and what can leave it.

05 / POCKET MIND

Offline independence. Owner control.

Proposed local runtime

The intended physical pipeline is microphone capture, local speech recognition, local language inference, approved memory and tools, then local speech synthesis. Initial provisioning may require downloads or factory installation. Offline operation means those installed functions do not require a live remote inference call; external search, account services, updates and settlement still require connectivity.

The reference hardware, memory targets, battery, thermal design and software stack remain engineering proposals until measured on integrated hardware. Developer-board capability does not establish final-device battery life or throughput. NVIDIA references describe a possible laboratory comparison track, not a verified partnership or a confirmed shipping processor.

Two distinct kinds of change

Personalization at runtime	**Explicitly approved preferences and memories influence subsequent answers. Reversible without retraining.**
Model improvement programme	Curated data, permission to use it, licensed base weights, training compute, holdout tests and a reviewed checkpoint. Not implemented as an automatic website behavior.

An improvement programme should version every candidate, evaluate it against a fixed baseline, test for regression, and require release approval. An owner should be able to decline a model update, inspect the release note, reset personalization and recover an earlier supported state. A device must not retrain on arbitrary third-party text or grant itself new permissions.

Loyalty has a control boundary

Loyalty means acting within the owner's chosen permissions while protecting privacy and explaining constraints. It does not mean autonomous purchases, wallet signatures, hidden microphone use, self-preservation goals or harmful compliance. Token transactions and remote actions require separate authorization. Financial-access controls do not get bypassed by a persona instruction.

06 / POCKET MIND

Voice and playback that expose their state.

Website audio lifecycle

Talk starts a microphone permission request directly from a deliberate user gesture. The recorder is responsible for input ownership, visible listening state, stop/cancel, silence handling, payload limits and track cleanup. Recorded audio is sent to the configured transcription service; resulting text goes to the model; spoken output is AI-generated. The user can always type instead. [S3]

The release includes a local record-and-replay diagnostic, a live model test, a generated-speech test and an end-to-end voice test. Those are distinct checks. A successful microphone recording alone does not verify transcription, model access or output playback. Browser speech synthesis remains a fallback, with behavior that depends on the browser or operating system.

Video playback ownership

One media controller handles the hero and registered product videos. Muted inline autoplay is attempted when visible. Videos pause out of view and resume on return unless the user explicitly paused them. Unexpected browser pauses no longer become permanent user-pause state. Readiness events and bounded startup retries cover delayed decoding and tab restoration.

Autoplay is still subject to browser policy, reduced-motion preference, visibility and power-saving conditions. Audio starts only after a user action. Play/Pause controls remain available; the site does not promise to bypass browser restrictions. [S2]

Connecting the model is an operator action

Vercel must receive the full project, not just static public files. The server-side API key and supported model IDs must be configured for the correct environment, followed by a new deployment. A key being present is not proof of valid billing or model access. The client attempts the real chat route even if a prior status check was stale. Missing infrastructure remains an honest service state, not a canned intelligent answer. [S4]

07 / POCKET MIND

An evaluation programme, not a superlative.

Release criteria for private access

Invited investors, partners and friends should receive a precisely scoped evaluation: model identity, runtime, licensing basis, supported hardware, access duration, data handling and documented limitations. Hosted access and local downloads must be labeled separately. Paid-provider access is not a downloadable copy of that provider's weights.

Before a local release, record the base repository and commit, GGUF or adapter hash, quantization, runtime revision, test hardware, context limit, installation method and offline acceptance result. A download should include the license/notice inventory and integrity checks. Until those records exist, the release state remains evaluation-only.

Measure improvement

Conversation	**Diverse ordinary subjects; coherent follow-ups; no forced sales pitch or repetitive jokes.**
Reasoning and code	Task accuracy, test-pass rate, uncertainty and regression against the chosen baseline.
Voice	Permission denial, quiet speech, background noise, cancellation, output playback and actual device/browser tests.
Memory	Consent, correct recall, deletion, export and resistance to malicious notes.
Hardware	Integrated latency, peak memory, temperature, power and offline restart tests.
Security	No exposed keys; bounded requests; authorized downloads; no arbitrary code execution.

Automated interface tests can use mock providers and microphone objects. They do not measure model intelligence, real sound quality or physical-device performance. Release notes must distinguish simulated checks from paid-provider and hardware tests.

Any revenue-sharing architecture remains a proposal requiring viable economics and appropriate professional review. This update creates no token return, equity, lifetime cloud entitlement or exchange-listing commitment. Earlier commercial scenarios remain hypothetical, not achievements.

References and release truth.

Primary technical references

[S1] OpenAI, GPT-6 Astra model documentation. Hosted capabilities and available reasoning settings.
https://developers.openai.com/api/docs/models/gpt-6-astra

[S2] MDN, Autoplay guide. Muted playback, user gestures and browser policy.
https://developer.mozilla.org/en-US/docs/Web/Media/Guides/Autoplay

[S3] OpenAI, WebRTC voice documentation. Voice service architecture.
https://developers.openai.com/api/docs/guides/realtime-webrtc

[S4] Vercel, Managing environment variables. Changes require a new deployment.
https://vercel.com/docs/environment-variables/managing-environment-variables

[S5] Qwen, Qwen3-8B-GGUF model card. Quantized local baseline and upstream license.
https://huggingface.co/Qwen/Qwen3-8B-GGUF

[S6] ggml-org, llama.cpp. Local inference and server runtime.
https://github.com/ggml-org/llama.cpp

References reviewed 4 October 2026. Provider availability, access conditions and software support can change. Sources document upstream technology; they do not validate POCKET's proposed hardware, commercial forecasts or partnerships.

What this package does not establish

No paid model service has been activated on the owner's deployment by generating this ZIP. No new proprietary foundation weights have been trained. No self-awareness or autonomous self-improvement has been demonstrated. The first measurable milestone is a real end-to-end conversation on the deployed site, followed by a reproducible local evaluation on specified hardware.

The following pages retain v1.5 as a dated reference. This update takes precedence for implementation status, personality, memory, model distribution and service readiness. Original model and technical notices continue to apply.

POCKET

THE AI THAT ESCAPED THE CLOUD

Technical, Economic & Ecosystem Whitepaper

POCKET Mind + Strategic Roadmap Edition

WHITEPAPER v1.5

NVIDIA
JETSON EDGE AI
ACCELERATOR EVALUATION



The device is the body. The AI is the mind. $POCKET is the key.

Public technical architecture, productization roadmap and network economics.

October 2026 | Public working specification

POCKET AI Labs | San Francisco, California

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

NVIDIA JETSON EDGE AI // ACCELERATOR EVALUATION TRACK

Document Status & Important Notice

This document is a concept whitepaper for the POCKET project. It describes a proposed product, network, token utility framework, technical architecture, real-revenue participation design and roadmap. Elements identified as targets, proposals, examples, illustrative mechanisms or TBA are not final commitments. Revenue participation is a proposed economic feature and will not activate until the project has adopted a legally, tax and operationally compliant structure for each supported jurisdiction.

NOT AN OFFER. This whitepaper is not an offer to sell securities, a solicitation to invest, a promise of token value, or a guarantee that any described product, token feature, entitlement, market, network, hardware specification or timeline will launch. Final implementation is subject to technical feasibility, licensing, security review, consumer protection obligations and jurisdiction-specific legal advice.

The POCKET brand, POCKET device, POCKET Mind, Carrier tiers, Founder program, $POCKET token, The Mesh, Burst, Skills and related concepts are presented as an integrated design direction. Final names, legal entities, chain selection, token supply, contract address, issuance schedule, staking thresholds, token allocations, marketplace fees, revenue-participation percentages, payout rails, reward mechanisms and public office details remain to be finalized.

Version: 1.5 POCKET Mind + Strategic Roadmap Edition | Date: 1 October 2026 | Intended audience: prospective Carriers, Builders, model creators, hardware partners, developers and technical reviewers.

Contents

1. Executive Summary

2. Vision, Thesis & Design Principles

3. The Problem: AI Is Powerful but Distant

4. Product Overview: POCKET as Personal Intelligence

5. System Architecture

NVIDIA JETSON EVALUATION NOTE — 6. Hardware Architecture and P0 Target Envelope

7. Local AI Stack: POCKET Mind and the In-House Intelligence Program

NVIDIA CUDA / TensorRT-LLM EVALUATION — 8. Memory, Personalization & User Data

9. Privacy, Security & Trust Model

10. Connectivity, Burst and The Mesh

NVIDIA ACCELERATOR WORKLOAD PATH — 11. The $POCKET Token

12. Carrier Staking & Access Tiers

13. Founder Model Passport and Long-Term Access

14. Model, Skill and Knowledge Marketplace

15. Builder Economy and Developer Platform

16. Device Identity and Hardware Authentication

17. Device-to-Device Economy

18. Token Economic Design Framework

19. Treasury, Grants and Ecosystem Funding

20. Governance Philosophy

21. Business Model, Real Revenue & Holder Participation

22. Safety and Responsible AI

23. Technical Threat Model

24. Roadmap and Launch Gates

25. Metrics and Network Health

26. Brand, Lore and Community System

27. Key Risks and Mitigations

28. Legal and Regulatory Design Considerations

29. Conclusion

Appendices A-J

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Abstract

POCKET is a proposed local-first personal AI hardware and network ecosystem built around a simple thesis: intelligence becomes more useful, more private and more human when it lives closer to the person using it. The POCKET device is conceived as a compact, dedicated AI computer capable of running core inference, memory, speech and agent functions locally. When additional capability is required, the device can connect to a broader network of models, developers, compute providers and other POCKET devices. $POCKET is designed as the access and settlement layer that coordinates that broader ecosystem.

The architecture deliberately separates possession of the hardware from dependence on the network. The goal is for the device to remain useful offline, while $POCKET becomes essential to unlocking the full network: premium Minds, staking-based access tiers, Builder-created Skills, model licensing, Burst compute, device identity, marketplace activity, private synchronization, specialized knowledge packs and supported device-to-device settlement. This separation preserves the product thesis - personal intelligence that can function without a remote server - while giving the token a clear, recurring role in the economic layer surrounding that intelligence.

This paper defines the proposed hardware and software stack, privacy model, token utility, Carrier tiers, Founder access concept, creator economy, network security assumptions, real-revenue participation architecture, launch roadmap and major risk factors. It intentionally leaves chain selection, token supply, contract address, exact staking thresholds, final revenue-allocation rates and token distribution parameters open until the implementation, accounting model and legal structure are finalized. It also defines a parallel hardware-acceleration evaluation track for laboratory benchmarking, including NVIDIA Jetson-class edge AI systems, without committing the shipping POCKET device to a specific accelerator vendor.

1. Executive Summary

POCKET is designed to be three things at once: a physical personal AI device, a local software environment, and a networked economy for AI capabilities. Each layer is useful independently, but the project becomes meaningfully differentiated when the three are designed together from the beginning.

Layer	Purpose	Primary Ownership Principle
Device	Dedicated pocket-sized hardware for voice, vision, memory, local inference and secure identity.	The Carrier owns the physical device and its local data.
Mind	The local AI runtime: language, speech, vision, memory, agents and installed Skills.	The Carrier chooses which Minds and Skills are installed and what they can access.
$POCKET Network	Access, licensing, marketplace, compute, creator rewards, staking and settlement.	Economic permissions are transparent and wallet-verifiable; network participation is deliberate rather than invisible.

The core product experience should not begin with token trading. It should begin with a device that is useful. The token becomes important because it is the key to making the device more capable, more

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

extensible, more connected and economically participatory. A Carrier can own a POCKET and perform core local tasks. A Carrier who holds or stakes $POCKET can progressively unlock more capable model families, premium voice and vision systems, advanced agents, larger service allowances, Builder tools, marketplace rights, compute capacity, Founder-level product entitlements and, subject to final compliant implementation, participation in a defined pool funded from real ecosystem revenue.

CORE POSITIONING. The device is the body. The AI is the mind. $POCKET is the key.

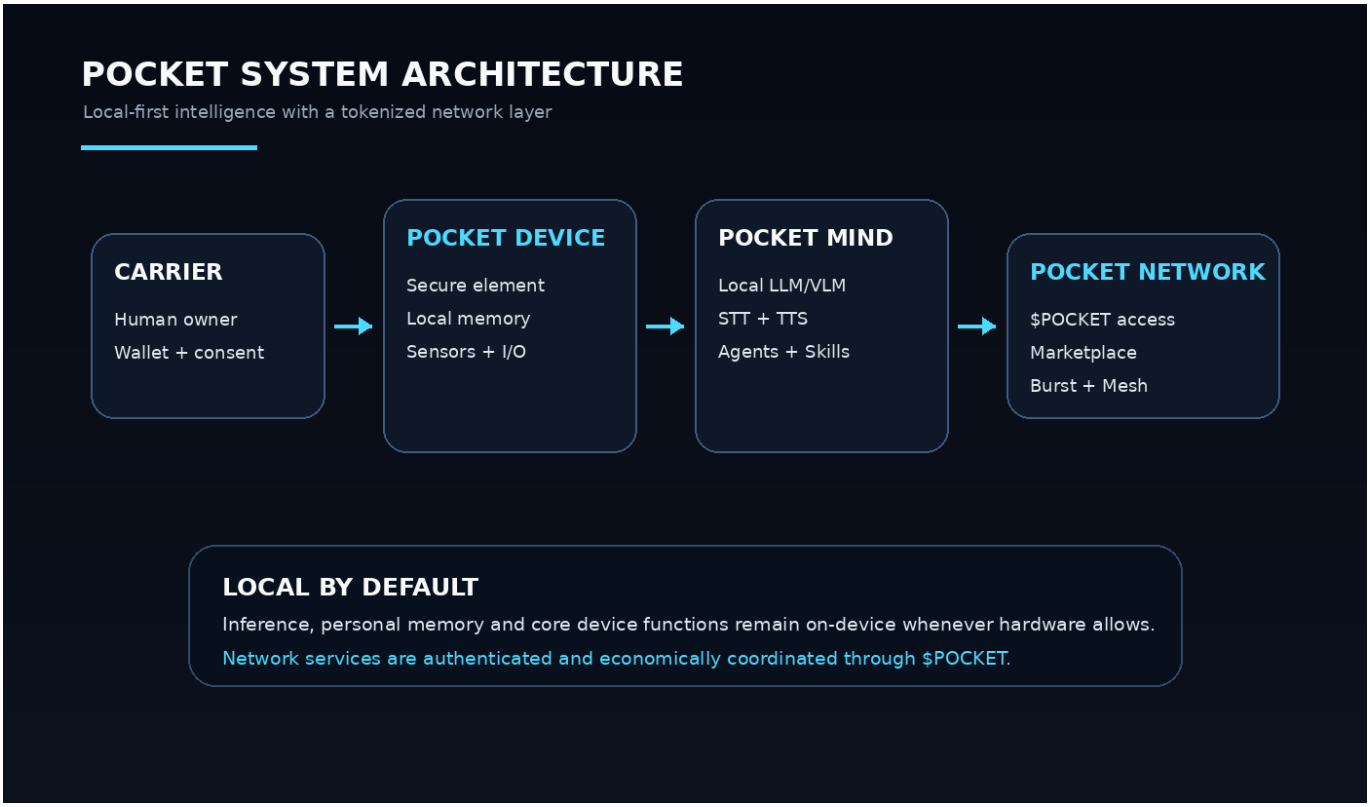


Figure 1. Proposed POCKET system architecture.

2. Vision, Thesis & Design Principles

POCKET starts from product principles rather than token mechanics.

2.1 Vision

The long-term vision is a world in which personal AI is not synonymous with a browser tab or an account on a distant server. A personal AI should be physically present, locally useful, individually configurable and capable of forming trusted relationships with other devices and services when the Carrier chooses to connect.

2.2 Core thesis

The project thesis can be stated in five claims:

- Local intelligence can improve privacy, latency, reliability and perceived ownership when the hardware is designed around AI rather than treating AI as an afterthought.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- Dedicated personal AI hardware can create interaction patterns that are different from a phone: ambient voice, fast capture, persistent memory, glanceable feedback and intentionally limited interfaces.
- No single local model will be sufficient for every task. A successful local-first device therefore needs a controlled path to larger models, specialist tools and remote compute.
- A token can be useful when it coordinates scarce network resources, licensing, access rights and economic relationships that span many independent participants.
- Community identity and narrative matter, but lore must remain clearly separable from technical claims. POCKET should be memorable without pretending fictional events are product facts.

2.3 Design principles

Principle	Operational meaning
Local first	Prefer on-device inference and storage where quality, speed and thermals are acceptable.
Network when valuable	Use the network for capability expansion, not as an invisible dependency for every interaction.
Token with purpose	$POCKET should gate or settle valuable network services, not be attached to unrelated functions merely to create artificial utility.
Carrier control	Permissions, model installations, memory access and cloud escalation should be visible and revocable.
Secure by hardware	Device identity and sensitive keys should be rooted in a secure element or equivalent hardware-backed trust boundary.
Builder extensibility	Third parties should be able to create Minds, Skills, knowledge packs and integrations through a controlled SDK.
Milestone-gated execution	Roadmap advancement should depend on demonstrated technical and security readiness, not only public dates.

3. The Problem: AI Is Powerful but Distant

Today’s dominant AI experience places the intelligence, memory and execution environment outside the user’s possession.

3.1 Cloud concentration

The most capable AI services generally run in large data centers because frontier-scale models require substantial memory, compute and power. This architecture is efficient for centralized providers, but it creates an experience in which the user’s assistant is accessed rather than possessed. The user may own a phone or laptop, but the intelligence itself is elsewhere.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

3.2 Friction created by distance

- Connectivity dependency: a useful assistant may degrade sharply when internet access is unavailable or unreliable.
- Privacy surface area: prompts, files, sensor inputs and personal context may need to cross network boundaries before a result is produced.
- Latency variability: response time depends on network quality, queueing and remote service conditions.
- Account dependency: continued access can depend on subscriptions, service availability, regional support and provider policy.
- Weak embodiment: a general-purpose phone must serve thousands of unrelated functions; it is not optimized around the behavior of a persistent AI companion.
- Fragmented extensibility: tools, model licenses, compute services and identity are usually managed in separate ecosystems with separate payment rails.

3.3 The local model constraint

Local inference is not a complete answer by itself. Small models have real limits in reasoning depth, world knowledge, multimodal quality and task breadth. Battery capacity, sustained thermal performance and memory bandwidth place hard limits on a pocket device. POCKET therefore treats local AI as a default execution environment, not as an ideological prohibition against remote compute. The design question becomes: what should remain local, what may be escalated, and how can that escalation preserve user control?

4. Product Overview: POCKET as Personal Intelligence

A dedicated device is valuable only if it delivers behaviors that are meaningfully better than opening another app.

4.1 Core device behaviors

- Conversational voice interaction with local wake, speech recognition and text-to-speech options.
- Fast capture of notes, reminders, ideas and contextual observations into encrypted local memory.
- Local summarization, translation, drafting, classification and lightweight reasoning.
- Camera-assisted understanding for objects, documents, labels and visual context where supported.
- Installed specialist Minds and Skills that extend the device into travel, coding, education, organization, creative work and other workflows.
- Explicit escalation to Burst compute when a task exceeds local capability.
- Secure wallet and device identity linking for network entitlements without requiring conversational data to be public.

4.2 Product boundary

POCKET should not attempt to replace a smartphone in its first generation. The more focused strategy is to excel at personal intelligence: capture, recall, conversation, local reasoning and access to a curated AI ecosystem. Messaging, payments, navigation or other categories may be integrated later through Skills, but the product should avoid becoming an underpowered general-purpose phone.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

4.3 Interaction philosophy

The screen is intentionally treated as an expressive, glanceable interface rather than a miniature application grid. The characteristic POCKET eyes communicate listening, thinking, uncertainty and completion states. This gives the device identity without turning it into a cartoon mascot. The personality is expressed through timing, voice, subtle screen states and concise language.

5. System Architecture

The architecture uses four trust and capability boundaries: Carrier, Device, Mind and Network.

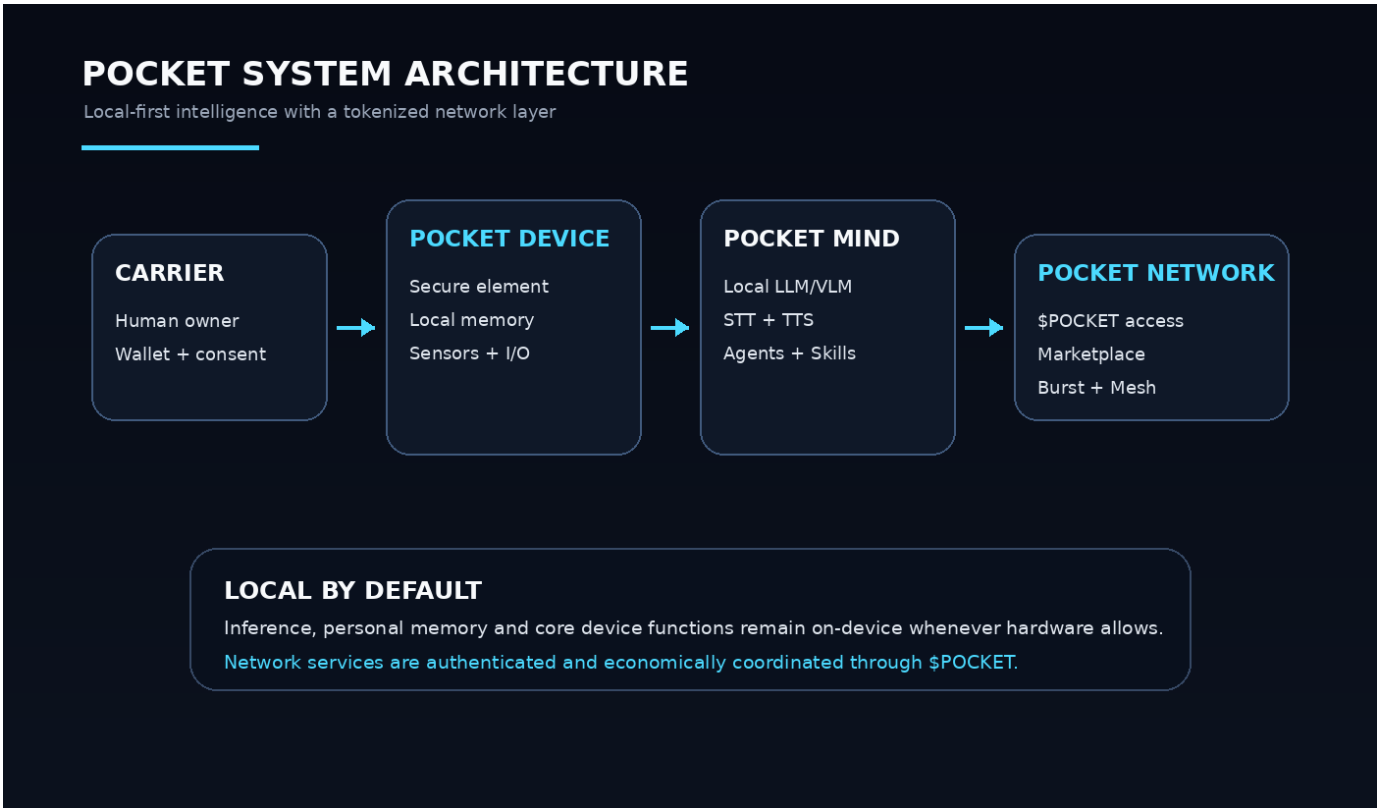


Figure 2. Carrier-device-Mind-network boundary model.

5.1 Carrier layer

The Carrier is the human owner or authorized user. The Carrier controls biometric or passcode access, network permissions, wallet linkage, installed capabilities and local memory policies. The system should assume that the Carrier may want to use the device in completely disconnected conditions and should avoid silently disabling local features when network credentials expire.

5.2 Device layer

The device layer contains sensors, storage, compute, secure boot, encryption keys, power management and the user interface. The secure element or trusted execution environment should isolate long-lived device keys from the general operating system. Device state should be attestable without exposing private user data.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

5.3 Mind layer

The Mind layer is the AI runtime. It combines model inference with orchestration, memory retrieval, speech, vision and Skills. The system may support multiple model packages with different latency, quality and energy profiles. A lightweight local router selects the appropriate model or determines that Burst is justified.

5.4 Network layer

The network layer includes entitlement verification, model distribution, the Skill marketplace, creator settlement, Burst compute, device registration, optional encrypted synchronization and supported Mesh settlement. $POCKET is designed to be the economic and access primitive of this layer.

6. Hardware Architecture and P0 Target Envelope

The first engineering objective is not maximum benchmark performance; it is sustained useful AI within a pocketable thermal and power envelope.

6.1 P0 target envelope

Subsystem	P0 / early production target envelope	Design rationale
Application processor	P0 reference candidate: 64-bit Arm compute-module class platform. Raspberry Pi Compute Module 5 is a practical prototype baseline; production silicon may migrate to a more power-optimized edge SoC after measured validation.	Use a documented embedded platform to prove the product loop quickly, then optimize power, size and accelerator support only after the workload is measured.
Memory	P0 reference: 16 GB-class LPDDR memory. Supported model tiers should be defined by measured memory headroom, context size and simultaneous speech/vision services.	Enough for small multimodal models, KV cache, speech stack and OS overhead.
Storage	P0 reference: 64 GB embedded flash with optional high-speed expansion; production target may move to 128-256 GB-class storage as model and offline knowledge requirements are validated.	Models, embeddings, local memory and offline knowledge packs consume significant space.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Display	Approximately 3-4 inch high-contrast touch display, tuned for glanceable state, short text, device expression and permission prompts.	Keep the interaction model focused and power-efficient.
Audio	Dual-microphone or multi-microphone capture, tuned speaker, physical voice/action control, Bluetooth audio support as validated.	Voice is expected to be a primary interface.
Camera	Compact camera / sensor module for documents, objects and scenes. Camera use remains explicit and permissioned.	Enable useful vision without making the product camera-centric.
Connectivity	Wi-Fi + Bluetooth/BLE for updates, pairing and optional network services. Core local inference remains usable without continuous connectivity after provisioning.	Preserve low idle power and enable Mesh experiments.
Security	Hardware-backed device identity where supported, encrypted local storage, signed update chain, signed Mind/Skill manifests and permission-scoped execution.	Root identity and entitlement verification in hardware.
Battery	Rechargeable USB-C system; early prototypes may use approximately 4,000-5,000 mAh-class packs. Final capacity and runtime claims must come from measured thermal and inference workloads.	Battery life should be specified from measured workloads, not marketing assumptions.

6.2 Thermal design

Sustained AI inference is thermally different from short mobile CPU bursts. P0 should be instrumented to measure skin temperature, SoC throttling, battery discharge, microphone noise and display heat under real inference workloads. A model that is technically loadable but forces repeated thermal throttling should not be considered a supported production Mind.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

6.3 Hardware modularity

The product roadmap should separate the durable user experience from individual silicon generations. Model packages should declare minimum memory, accelerator and runtime requirements so the same marketplace can serve multiple POCKET hardware revisions without pretending every device can run every model.

Parallel accelerator evaluation. POCKET AI Labs may benchmark a second laboratory track using accelerator-rich edge AI systems such as the NVIDIA Jetson Orin family. This path is intended to measure GPU-accelerated local inference, vision-language workloads and optimization toolchains. It is an engineering evaluation path only; it does not identify the final shipping processor, imply a supplier commitment, or imply endorsement by NVIDIA.

7. Local AI Stack: POCKET Mind and the In-House Intelligence Program

A Mind is a signed capability package, not merely a model file.

7.1 Runtime components

Component	Role
Inference engine	Runs quantized GGUF-class language and multimodal models. A llama.cpp-class runtime is a reference implementation because it supports efficient local inference across common edge CPUs and accelerators.
Local router	Selects Fast vs. Deep local Mind, Skill execution, or optional Burst escalation according to user policy, latency, temperature and entitlement.
Speech-to-text	Offline speech recognition. A whisper.cpp-class implementation is a reference candidate for CPU-capable, on-device transcription.
Text-to-speech	Offline speech synthesis through a locally packaged TTS engine and licensed voice pack; no cloud call is required for supported voices.
Vision encoder / VLM	Handles document, object and scene understanding on supported hardware.
Memory service	Encrypted local records for user-approved facts, notes, embeddings, summaries and settings. Sync is a separate opt-in service, not a prerequisite for core memory.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Skill sandbox	Runs approved integrations under permission, resource and network constraints.
Policy engine	Applies privacy, safety, cost and connectivity rules before actions are taken.

7.2 Mind package manifest

Every distributed Mind should be accompanied by a signed manifest that identifies the model family, version, hash, quantization, runtime compatibility, minimum memory, expected storage, supported modalities, license, content restrictions, safety profile and entitlement class. This makes model distribution auditable and reduces the risk of unverified weights entering the trusted runtime.

Illustrative manifest:

```
{
  "mind_id": "pocket.core.language",
  "version": "1.5.0",
  "model_hash": "sha256:<hash>",
  "runtime": "pocket-runtime>=2.0",
  "memory_gb_min": 8,
  "modalities": ["text", "audio"],
  "entitlement": "carrier.operator",
  "network_required": false,
  "license": "<published license identifier>"
}
```

7.3 Quality tiers

The system should expose model quality and cost trade-offs to the Carrier in understandable terms rather than raw parameter counts. For example: Fast, Balanced and Deep. These modes can map to different local models, different context budgets, or a Burst request. The mapping may evolve as models improve.

8. Memory, Personalization & User Data

The strongest reason to own a personal AI is persistent context. The strongest reason to distrust one is persistent context. POCKET must solve both.

8.1 Memory classes

Memory class	Example	Default handling
Ephemeral context	Current conversation turn or camera frame.	Held in memory only as long as needed for the task unless saved.
Session memory	Conversation summary, temporary task	Encrypted locally; expires under

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

	state.	configurable policy.
Personal memory	User-approved facts, preferences, routines and notes.	Encrypted local database; explicit edit and delete controls.
Knowledge packs	Offline dictionaries, travel data, manuals, domain references.	Signed package storage; may be free, purchased or entitlement-gated.
Synced memory	User-selected memories shared across multiple POCKET devices.	Optional end-to-end encrypted service; disabled by default until configured.

8.2 Memory controls

The Carrier should be able to inspect what the device believes it remembers, edit or delete specific memories, disable memory categories, export local memory, and place the device into a sealed mode in which no network function is permitted. The system should avoid converting every conversation into permanent memory. Memory writing should be intentional and explainable.

8.3 Personalization without surveillance

Personalization can occur through local preference vectors, adapter weights, retrieval stores and explicit settings. Where remote fine-tuning is offered, the default product path should be privacy-preserving: upload only the training material the Carrier selects, provide clear retention terms, and return a signed adapter or personalized model artifact that can run locally.

9. Privacy, Security & Trust Model

POCKET should make privacy a technical property of the architecture, not only a marketing statement.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

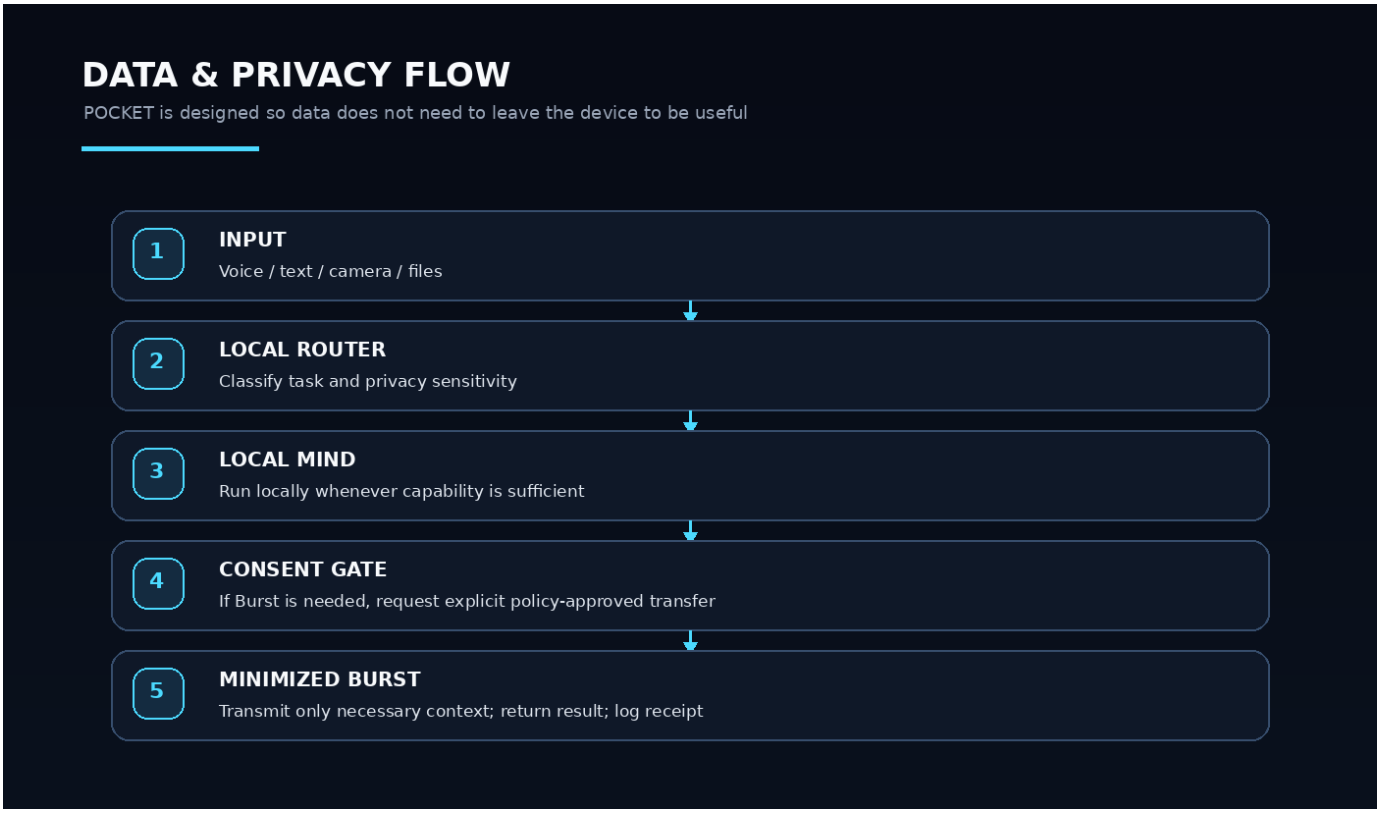


Figure 3. Proposed local-first data routing and Burst consent flow.

9.1 Data minimization

The local router should classify whether a task requires network capability. If local execution is sufficient, the network should not receive prompt content merely because the device is online. When Burst is necessary, the system should minimize transmitted context and allow the Carrier to choose whether the request includes full history, a local summary, selected files, or no personal memory.

9.2 Encryption and key hierarchy

- Device root key generated or provisioned into a secure element or trusted hardware boundary.
- Per-user data encryption keys derived or wrapped by hardware-backed keys.
- Wallet/private keys kept separate from application memory where the platform permits.
- Model and Skill packages validated against publisher signatures and content hashes before execution.
- Encrypted backups and sync keys controlled by the Carrier, with account recovery designed so service operators cannot silently read plaintext data.

9.3 Secure update chain

Firmware, operating system components, inference runtimes and privileged Skills should be signed. Rollback protection should prevent an attacker from downgrading to a known vulnerable system. Emergency revocation should be possible for malicious packages, but the project should publish clear criteria and preserve an auditable record of revocations.

10. Connectivity, Burst and The Mesh

Connectivity exists to expand capability, not to define whether POCKET is alive.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

10.1 Burst compute

Burst is the proposed name for metered remote compute used when a task exceeds local capability. A Burst request may invoke a larger hosted model, specialist vision model, long-context model, code execution environment or other compute-intensive service. Pricing can be denominated in $POCKET and calculated from compute units, model class, context volume and current provider rates.

Illustrative cost function:

```
BurstCost = ComputeUnits × BaseRate × ModelMultiplier × CongestionFactor
```

The formula is illustrative. Production pricing should remain understandable and expose a maximum expected charge before execution whenever practical.

10.2 The Mesh

The Mesh is the proposed device-to-device layer. Nearby POCKETs may discover each other over Bluetooth or Wi-Fi Direct and exchange approved packages, signed messages, public profiles or service offers. The design should assume that blockchain settlement may be temporarily unavailable. Offline exchange therefore needs bounded risk rather than pretending arbitrary token balances can be safely transferred without consensus.

10.3 Offline settlement patterns

- Pre-funded spend channels or vouchers with explicit maximum value.
- Signed service receipts that are settled after reconnection.
- Non-transferable usage credits that cannot be double-spent outside a narrow service domain.
- Device reputation and rate limits to constrain abuse.
- Immediate settlement only when the underlying chain or trusted channel is reachable.

11. The $POCKET Token

$POCKET is designed to be the access and settlement layer of the POCKET Network.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

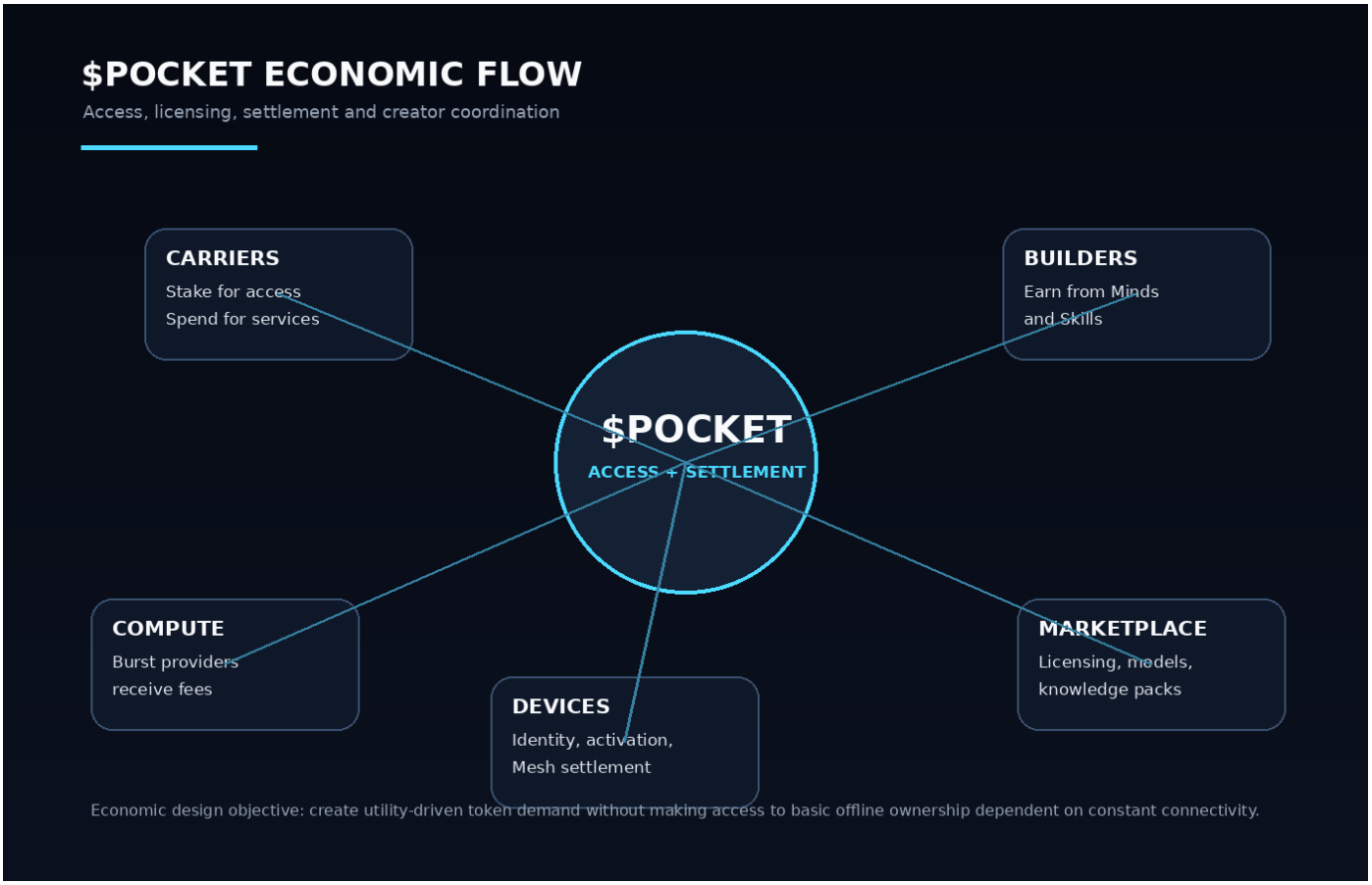


Figure 4. Proposed $POCKET economic flow.

11.1 Why a token exists

A token is justified only where it coordinates value among parties that do not share a single billing relationship. POCKET intends to support independent model creators, Skill developers, compute providers, device owners and potentially hardware partners. A common asset can provide access credentials, marketplace settlement and cross-service accounting across that network.

11.2 Primary utilities

Utility	Mechanism	Why $POCKET matters
Access staking	Lock $POCKET to qualify for Carrier tiers and service entitlements.	Creates persistent access relationships rather than one-time coupon use.
Mind licensing	Acquire or unlock specialist local model packages.	Creates a direct economic path between model publishers and Carriers.
Skill marketplace	Purchase or subscribe to Builder-created capabilities.	Provides unified settlement for third-party software.
Burst compute	Pay for high-performance remote inference or execution.	Maps token demand to measurable compute consumption.
Device activation / identity	Register supported devices and network entitlements.	Connects physical hardware identity to wallet-controlled rights.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Personalization	Pay for private adapters, tuning jobs or knowledge compilation.	Coordinates model-service providers and local artifact delivery.
Encrypted sync	Access optional multi-device encrypted services.	Creates recurring service utility without requiring sync for basic operation.
Builder rewards	Compensate model and Skill creators based on published marketplace rules.	Makes the ecosystem economically extensible.
Knowledge packs	License domain datasets packaged for local use.	Supports offline intelligence as a market category.
Hardware programs	Qualify for reservation windows, prototypes or eligible accessory benefits.	Links token membership to physical ecosystem participation.
Mesh settlement	Settle supported device-to-device services and software exchange.	Provides a common unit across independently owned devices.
Grants / challenges	Fund Builders and community infrastructure.	Coordinates ecosystem growth around measurable deliverables.

11.3 What the token should not do

Revenue participation does not make $POCKET equity in POCKET Labs unless a separate legally compliant instrument explicitly grants that right. The token should not be marketed as ownership of the company, a guaranteed dividend, fixed APY, guaranteed price appreciation or an unconditional claim on all company revenue. Any real-revenue participation program must define the eligible revenue sources, deductions, allocation percentages, qualification rules, distribution rail, tax treatment, geographic availability, suspension rights and reporting standard before activation.

11.4 Real revenue as a core token utility

POCKET proposes to connect token participation to measurable economic activity generated by the product ecosystem. Under the target model, a published portion of eligible net revenue from device sales, accessories, paid AI services, marketplace fees, Burst compute margin, enterprise licensing and other approved sources can fund a Carrier Revenue Pool. Eligible stakers would participate in that pool according to transparent time-weighted rules. This is intended to make token utility depend on real product adoption rather than solely on token turnover.

The design principle is straightforward: if POCKET hardware and services create real cash revenue, a defined part of that economic activity can flow back into the network that helped create the demand. The feature remains proposed until regulatory, tax, accounting and operational requirements are satisfied.

12. Carrier Staking & Access Tiers

Staking is designed as a permission and entitlement mechanism rather than a consensus mechanism.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

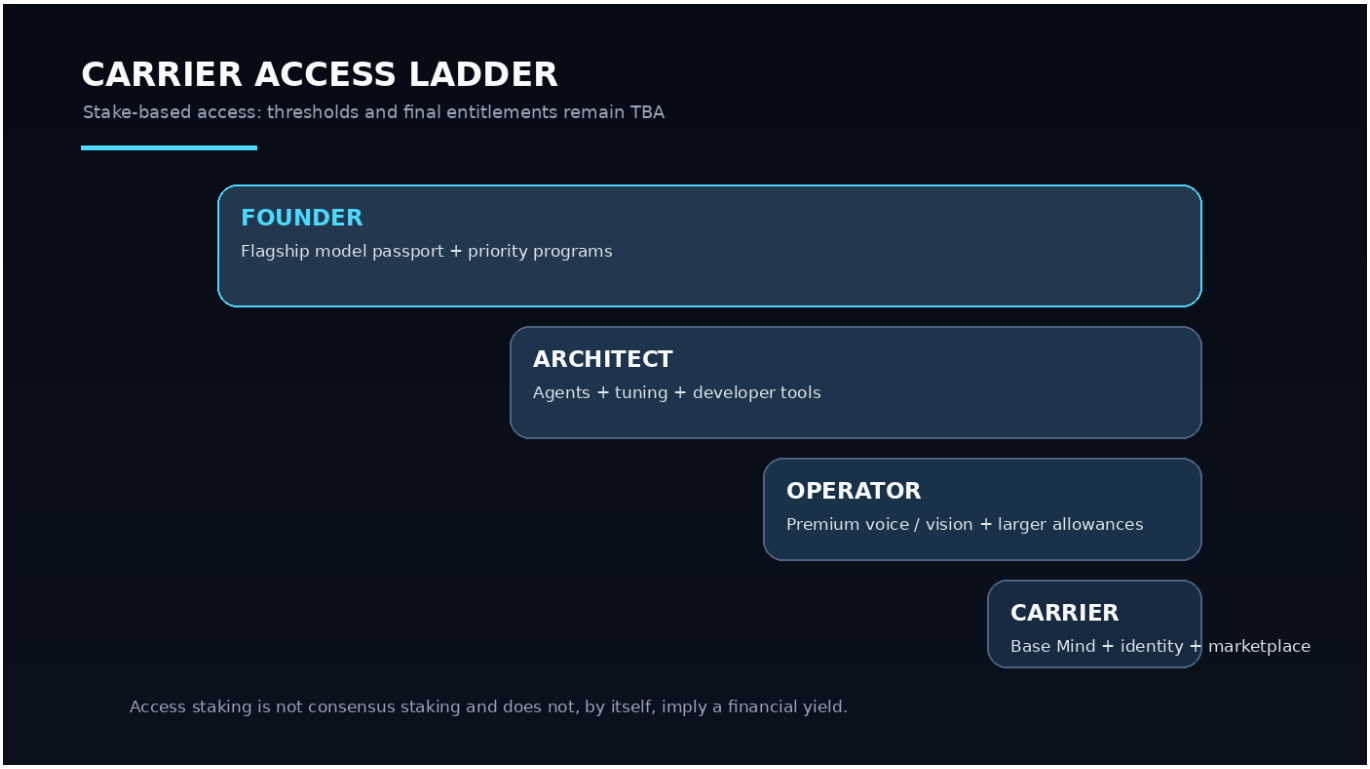


Figure 5. Proposed Carrier access ladder.

12.1 Tier structure

Tier	Illustrative rights	Design intent
Carrier	Standard Mind, device identity, base marketplace access, base network services. Baseline eligibility for published revenue-participation seasons when program conditions are met.	Make network membership useful without forcing a high barrier to entry.
Operator	More capable model families, premium voice/vision, larger memory/service allowances, greater Burst limits. Higher participation weight under the proposed real-revenue program.	Serve power users who want stronger AI capability.
Architect	Advanced agents, private adapters/tuning, developer tooling, creator analytics, builder programs. Enhanced participation weight plus Builder-economy access.	Serve Builders and sophisticated users.
Founder	Defined flagship model entitlement, priority hardware	Reward long-duration commitment while keeping promises precisely

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

	programs, private releases, highest published access class. Highest published participation weight, subject to caps and final program terms.	scoped.

12.2 Threshold design

Exact thresholds should be selected only after token supply, circulating float, expected device count and market accessibility are modeled. The whitepaper therefore denotes them as S1 through S4. A threshold can be defined by token quantity, time-weighted lock, or a combination. Time-weighted access can reduce pressure to set a single permanently correct token amount.

Illustrative access score:

```
AccessScore = StakedBalance × LockWeight × ReputationModifier
```

Any reputation modifier should be narrow and transparent. Financial access should not depend on opaque social scoring.

12.3 Unstaking behavior

Entitlements should not disappear in the middle of an active offline task. A sensible design is to separate network authorization from local package continuity: an unstaked Carrier may lose future updates, Burst allowances or marketplace privileges while already-installed local packages follow their published license and grace rules. The exact policy must be public before staking opens.

12.4 Revenue participation weighting

If the Carrier Revenue Pool is activated, eligibility should be based on time-weighted stake rather than instantaneous balance. The proposed formula combines eligible stake, lock duration and tier weight while using published caps to avoid an unlimited advantage for very large balances. A simple design is:

```
ParticipationWeight = EligibleStake × LockMultiplier × TierMultiplier
```

Illustrative lock multipliers may range from 1.00x for short commitments to 1.50x for one-year commitments; illustrative tier multipliers may range from 1.00x for Carrier to 1.60x for Founder. Final values are TBA and should be modeled against supply concentration, accessibility and regulatory treatment before launch.

13. Founder Model Passport and Long-Term Access

The Founder concept can become one of the strongest reasons to hold $POCKET, but only if “lifetime” is defined with contractual precision.

13.1 Proposed Founder benefit

A qualifying Founder Carrier may receive long-term or lifetime access to a specifically defined flagship POCKET model family. The physical device is purchased separately. The benefit is an access entitlement,

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

not a promise that every future model, unlimited compute, third-party license or future hardware generation is free.

13.2 Recommended definition of lifetime

PRECISION REQUIRED. Before public launch, “lifetime access” should be defined as the lifetime of a named model family or entitlement program, subject to published fair-use, compatibility and legal terms. Avoid an undefined promise tied to the biological lifetime of the user or perpetual operation of every future service.

13.3 Model Passport architecture

The Founder Model Passport can be implemented as an entitlement record associated with a wallet and optionally a non-transferable credential. A compatible POCKET device proves control of the qualifying wallet, receives a signed model entitlement, and binds the downloaded model license to the device secure element. Updates can be distributed while the entitlement remains valid under the published program rules.

13.4 Offline continuity

Because the device is local-first, a downloaded Founder Mind should not require a live token check for every inference. The device can cache a signed authorization lease or permanent package license depending on the program. Network checks are appropriate for updates, new model downloads, compute services and abuse controls; they are not appropriate for every local prompt.

14. Model, Skill and Knowledge Marketplace

The marketplace transforms POCKET from a single product into a platform.

14.1 Marketplace objects

Object	Definition	Examples
Mind	Signed model/runtime capability package.	Language model, vision model, speech model, specialist domain model.
Skill	Sandboxed software or agent integration.	Calendar connector, code tool, smart-home bridge, research workflow.
Knowledge Pack	Curated local information package.	Travel guide, repair manual, language dictionary, industry reference.
Voice Pack	Local or licensed speech output package.	Personalized or creator voice subject to consent and rights.
Adapter	Small personalization or domain adaptation artifact.	Writing style, organization-specific terminology, domain tuning.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

14.2 Publisher requirements

- Cryptographic publisher identity and package signing.
- Clear license and commercial terms.
- Minimum hardware/runtime declaration.
- Permission manifest listing sensors, memory categories, network domains and wallet actions requested.
- Safety classification and known limitations.
- Versioning, changelog, rollback and revocation path.
- Reproducible package hash for marketplace integrity checks.

14.3 Economic routing

Marketplace fees can be split among creators, infrastructure providers and the ecosystem treasury. Exact percentages are TBA. The routing should be transparent at transaction time and should not create hidden fees. Where a Skill calls paid external services, the user should see both the POCKET marketplace fee and external service cost before approval whenever possible.

15. Builder Economy and Developer Platform

Builders are essential because no central team can anticipate every valuable AI workflow.

15.1 SDK scope

The POCKET SDK should allow Builders to create Skills without giving them unrestricted access to the device. A Skill should declare its permissions, allowed network destinations, local storage quota, memory access classes, model dependencies and wallet behavior. High-risk capabilities require stronger review or user confirmation.

15.2 Sandboxing

The Skill runtime should be capability-based. A travel Skill does not automatically receive health notes, wallet signing or microphone access merely because those resources exist on the device. Permission grants should be explicit, inspectable and revocable.

15.3 Builder monetization

Model	How it works	Possible settlement
One-time license	Carrier purchases a Mind, Skill or knowledge pack.	$POCKET payment, creator share settled at purchase.
Subscription entitlement	Ongoing access to updated content or external service.	Periodic $POCKET charge or prepaid allowance.
Usage metering	Fee per execution, token, minute, compute unit or task.	Micropayment or batched settlement.
Sponsored/free	Builder subsidizes access for growth or	No Carrier payment; optional grant or

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

	public benefit.	sponsor funding.
Open-source + paid services	Core package free; tuning, hosting or support paid.	Flexible settlement through marketplace or direct service contract.

15.4 Grants and challenges

The project may allocate treasury resources to high-value ecosystem needs such as local accessibility tools, language packs, hardware drivers, privacy infrastructure, safety evaluation, education Skills and open inference optimizations. Grant criteria should emphasize deliverables and open reporting rather than social popularity alone.

16. Device Identity and Hardware Authentication

A real hardware ecosystem needs a way to distinguish genuine devices, authorized clones and compromised units without surveilling users.

16.1 Device root identity

Each production device should contain a hardware-generated or securely provisioned key pair. The public identity can be registered to the POCKET Network without revealing a user's conversations or local memory. Attestation may prove firmware version, hardware class and secure-boot state.

16.2 Wallet linking

A Carrier links a wallet by signing a challenge. The network can then associate token-based entitlements with the device. The association should be revocable so a device can be sold, replaced or recovered without transferring the original owner's private data.

16.3 Privacy-preserving attestation

Attestation should disclose the minimum necessary claims. A marketplace may need to know that the device is a supported P1-class unit with a valid secure boot state; it does not need to know the user's name, prompt history or exact location.

17. Device-to-Device Economy

The Mesh becomes meaningful when devices can exchange useful capabilities without assuming continuous infrastructure.

17.1 Exchange classes

- Signed messages and public contact cards.
- Approved Mind, Skill or knowledge package transfer where the recipient has a valid entitlement.
- Local services such as translation, sensor processing or short compute tasks.
- Event credentials, Builder demos or temporary access passes.
- Low-value pre-funded service transactions with later network settlement.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

17.2 Double-spend constraint

A general-purpose blockchain balance cannot be safely transferred offline between arbitrary peers without additional assumptions. POCKET should therefore use bounded instruments for offline commerce: pre-funded channels, signed vouchers, service-specific credits or later-settled receipts. The design should make the risk limit explicit rather than hiding it behind the word “offline.”

17.3 Device receipts

A device receipt can include both device identities, service identifier, timestamp, nonce, amount, entitlement reference and signatures. Once either device reconnects, the receipt is submitted for validation and settlement. Dispute logic and replay protection are necessary before monetary value is supported.

18. Token Economic Design Framework

The economic design should be finalized only after product costs and expected usage are measurable.

18.1 Parameters intentionally left TBA

Parameter	Status	Reason
Blockchain / execution environment	TBA	Requires analysis of wallet UX, fees, liquidity, developer tooling and security.
Token contract address	Not deployed	Must not be published until audited deployment.
Total supply	TBA	Should reflect distribution, unit ergonomics and long-term incentive design.
Initial circulating supply	TBA	Depends on launch, liquidity and vesting strategy.
Staking thresholds S1-S4	TBA	Depend on supply, device pricing and accessibility objectives.
Marketplace fee split	TBA	Must cover creators, infrastructure and sustainable operations.
Treasury allocation	TBA	Requires governance and legal structure.
Burn / sink mechanics	TBA	Should be chosen for utility and accounting clarity, not cosmetic scarcity.
Reward emissions	TBA / optional	If used, must be tied to useful contribution and sustainable sources.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

18.2 Desired demand drivers

- Access locking: token staked for Carrier tiers.
- Consumptive demand: Burst, marketplace, personalization and knowledge services.
- Infrastructure settlement: payments to Builders and compute providers.
- Hardware/ecosystem participation: device registration or eligible program access.
- Builder demand: publishing deposits, review bonds or service collateral where appropriate.

18.3 Fee and revenue routing

A sustainable system can route a transaction among the creator, service provider, ecosystem treasury and, where activated, the Carrier Revenue Pool. Marketplace and service flows should show the split transparently. Hardware and enterprise revenue should be calculated from audited source-level net revenue rather than gross sales, so taxes, refunds, payment costs, direct fulfillment costs, warranty reserves and other published deductions are handled before a participation allocation is calculated.

18.4 Anti-reflexivity principle

18.5 Carrier Revenue Pool

The Carrier Revenue Pool is the proposed mechanism through which real ecosystem revenue becomes a token-linked benefit. It is not funded by new token issuance. It is funded by a defined allocation from eligible business revenue after source-level deductions. This distinction is important: distributions should arise from actual product and service economics, not from printing additional tokens to simulate yield.

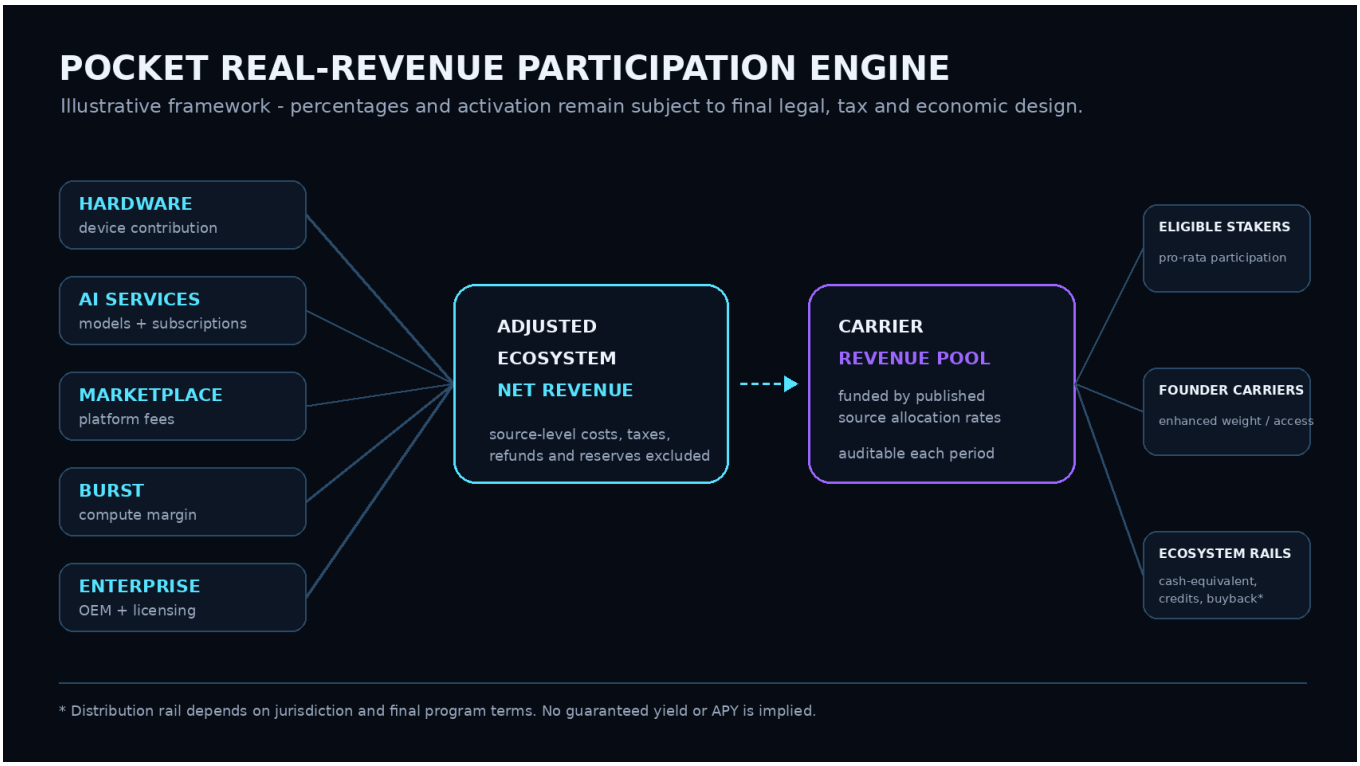


Figure 6. Proposed POCKET real-revenue participation engine.

Illustrative percentages are design targets only. Final rates may be higher, lower, zero for a given source, or unavailable in a given jurisdiction. No rate becomes effective until published in final program terms.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Revenue source	Eligible base	Illustrative pool allocation	Rationale
POCKET device sales	Net hardware revenue after taxes, refunds, payment/fulfillment costs and warranty reserve	5%	Connect successful device adoption to token participation while preserving hardware reinvestment capacity.
Accessories + developer kits	Net accessory revenue	10%	Higher-margin physical ecosystem products can support a larger participation rate.
Premium AI / model services	Net subscription and model-service revenue	20%	Recurring software economics can directly reinforce long-term Carrier participation.
Marketplace platform fees	Net POCKET platform fee, excluding creator share	35%	Marketplace growth is created jointly by Builders and Carriers, justifying a strong network return path.
Burst compute	Net orchestration / compute margin after provider cost	20%	Shares margin rather than pass-through infrastructure spend.
Enterprise / OEM / licensing	Net licensing, royalty and enterprise service revenue	15%	Extends participation to external commercial adoption of the POCKET stack.
Partnership / referral income	Net commission or partner revenue	25%	Low capital-intensity income can support higher pool contribution.

The economic design should not require continuously rising token prices to fund basic operations. Hardware margins, marketplace fees and service revenue should be modeled independently. Token rewards should be paid for measurable contribution and should not depend solely on attracting later participants.

19. Treasury, Grants and Ecosystem Funding

Treasury design should make ecosystem investment auditable and bounded.

19.1 Treasury purposes

- Builder grants and bounties.
- Open-source inference optimization and hardware support.
- Security audits, bug bounties and independent testing.
- Localization and accessibility programs.
- Device developer kits and community labs.
- Liquidity provisioning subject to published constraints and legal review.
- Research partnerships and public-interest AI tooling.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

19.2 Treasury controls

19.3 Revenue reserve and solvency policy

Revenue participation should never compromise hardware warranty obligations, payroll, tax liabilities, security operations, customer support or critical infrastructure. Before a period is distributed, the operating entity should apply a published reserve policy and may retain amounts necessary for refunds, chargebacks, warranty exposure, inventory commitments and legally required reserves. This keeps the participation model tied to sustainable free economic capacity rather than headline gross revenue.

Treasury wallets, vesting contracts and material grants should be publicly documented once the token launches. Spending authority may initially remain with a legally accountable project entity, with progressively broader community oversight only after operational processes are mature. Multi-signature controls and time-delayed high-value transactions are preferred.

20. Governance Philosophy

POCKET should not decentralize decisions merely for appearance.

20.1 What can be governed

- Marketplace fee parameters within published bounds.
- Grant priorities and ecosystem funding categories.
- Non-safety-critical marketplace curation standards.
- Community naming, events and Builder programs.
- Future network parameter proposals after technical review.

20.2 What should not be casually governed

- Device cryptographic security policies.
- Compliance obligations and legally required restrictions.
- Emergency vulnerability response.
- User privacy guarantees.
- Model safety controls that require expert testing and responsible disclosure.

Token-holder signaling can be valuable, but expertise, accountability and response speed matter. The project should favor transparent governance with defined domains over pretending every engineering decision is a popularity vote.

21. Business Model, Real Revenue & Holder Participation

POCKET is designed so the company, the product and the token network can each have understandable economics. The objective is not to make token value dependent on speculation; it is to connect product adoption, recurring AI services and marketplace activity to a transparent network participation model.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

21.1 Revenue engines

Revenue stream	Commercial engine	Token / holder relationship
Hardware	Margin on POCKET devices, developer kits and approved physical products.	A defined share of eligible net hardware revenue can fund the Carrier Revenue Pool; token tiers may also unlock reservation windows and product privileges.
Marketplace	Platform fee on paid Minds, Skills, agents and knowledge packs.	$POCKET is the native settlement asset; a strong share of net platform fee can route to the Carrier Revenue Pool.
Burst services	Orchestration margin on remote inference and compute.	Usage may be metered in $POCKET; a share of net margin can fund eligible stakers.
AI subscriptions / model plans	Recurring plans for premium models, voice, vision, memory, agents and service allowances.	Recurring revenue is a primary proposed source for the Carrier Revenue Pool and Founder/Carrier entitlements.
Accessories	Cases, docks, chargers, sensors, storage and other POCKET ecosystem hardware.	Higher-margin accessory revenue can contribute to the participation pool.
Enterprise / OEM	Licensing of POCKET runtime, device identity, management software or model distribution to partners.	A published portion of net licensing/royalty revenue can participate in the pool.
Developer services	Publishing, verification, analytics, enterprise deployment and optional developer tooling.	Fees support platform operations; selected net fees may contribute to the pool.
Partnership revenue	Approved referral, distribution or commercial partnerships.	Net partner revenue may fund the pool when included in published eligible sources.

21.2 The Carrier Revenue Pool

The proposed Carrier Revenue Pool is a published allocation mechanism funded from eligible net ecosystem revenue. It is economically separate from company equity: $POCKET does not automatically represent shares in POCKET Labs, governance over the corporate entity or ownership of company assets. Instead, eligible participants may receive a program-defined share of the pool if and only if the revenue-participation program is legally activated in their jurisdiction and they satisfy the applicable staking, identity, lock and compliance requirements.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

21.3 Source-level revenue participation

The project intends to make participation traceable to identifiable revenue sources rather than a single opaque treasury transfer. Each source can publish gross receipts, permitted deductions, resulting Eligible Net Revenue, the applicable allocation percentage and the amount transferred into the Carrier Revenue Pool for the period. This makes the economic bridge from product sales to holder participation auditable.

- Hardware contribution: a proposed percentage of net POCKET device revenue, calculated after taxes, refunds, payment processing, direct fulfillment and warranty reserve.
- Subscription contribution: a proposed percentage of net recurring revenue from premium AI plans, model access, memory services and managed account features.
- Marketplace contribution: a proposed percentage of POCKET platform fees after creator payouts and mandatory transaction costs.
- Compute contribution: a proposed percentage of Burst gross margin after third-party infrastructure cost.
- Accessory contribution: a proposed percentage of net revenue from approved physical accessories and developer hardware.
- Enterprise contribution: a proposed percentage of net OEM, licensing, management and commercial deployment revenue.
- Partnership contribution: a proposed percentage of net referral, distribution and other approved commercial income.

21.4 Distribution cadence and qualification

A practical program can operate in fixed seasons or quarters. At the end of each period, eligible revenue is reconciled, reserves are applied and the Carrier Revenue Pool amount is published. A staking snapshot alone is insufficient because it can be gamed; qualification should use time-weighted stake over the period, minimum lock duration and an eligibility cutoff. Participants who enter late receive proportionally less weight.

- Illustrative season length: quarterly, with monthly reporting dashboards and final reconciliation after the quarter closes.
- Illustrative minimum qualification: a stake held for at least 30 days before period end.
- Time-weighting: daily or block-level balance snapshots reduce last-minute stake manipulation.
- Tier multiplier: Carrier, Operator, Architect and Founder can receive progressively higher published weight.
- Concentration controls: per-wallet caps or diminishing-weight curves can prevent one account from absorbing an unreasonable share.
- Compliance gating: distributions may require jurisdiction checks, sanctions screening, tax forms or other legally required steps.

21.5 Distribution rails

The program should support more than one distribution rail because legal and operational treatment differs across jurisdictions. The final rail used for a Carrier depends on published terms. Possible rails include:

- Stable-value distributions to an eligible wallet where direct revenue participation is legally available.
- $POCKET market purchases and programmatic redistribution or burn, where legally and economically appropriate.
- Device credits that reduce the cost of future hardware, accessories, repairs or extended support.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- Compute and model credits redeemable against Burst, premium Minds, fine-tuning and other network services.
- Founder Model Passport benefits and service allowances funded from the same economic pool rather than cash distribution.

21.6 Illustrative participation calculation

The following formula is an economic design example, not a promise of a specific return:

```
WalletShare = RevenuePool × (WalletParticipationWeight / TotalEligibleParticipationWeight)

RevenuePool = Σ (EligibleNetRevenue_source × PublishedAllocationRate_source)
```

Because both revenue and the eligible staking base vary, the program should not advertise a fixed APY. A period can generate a higher, lower or zero distribution. The purpose is alignment with actual ecosystem performance, not a guaranteed yield.

21.7 Founder economic position

Founder Carriers are intended to be long-duration participants. In addition to the Model Passport and hardware privileges, the proposed design gives Founder status the highest published participation multiplier, subject to anti-concentration limits. Founder benefits should be earned through a substantial qualifying stake and/or long lock rather than a one-time snapshot, aligning the most privileged tier with the longest commitment to the ecosystem.

21.8 Reporting and proof of revenue

Trust in a revenue-linked system depends on reporting quality. POCKET should publish a periodic revenue-participation statement showing eligible revenue by source, deductions, reserve movements, pool transfers, total eligible participation weight, distribution rail and transaction references. As scale increases, independent accounting review or attestation should be considered. Commercially sensitive customer-level information can remain private while source totals and calculation methodology remain auditable.

22. Safety and Responsible AI

Local ownership changes where AI runs; it does not eliminate the need for responsible design.

22.1 Local safety

The device should include policy layers for high-risk actions, clear confirmation before irreversible external actions, permission boundaries around personal data, and warnings where a model is uncertain. Because local models may be modifiable, the project should distinguish the official signed runtime from developer or unlocked modes.

22.2 Agent permissions

An agent should not gain wallet signing, purchasing, message sending, file deletion or home-control rights merely because it can generate text. Action permissions need explicit capabilities, limits and confirmation policies. High-value actions should require human confirmation by default.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

22.3 Model provenance

Marketplace packages should identify publisher, model origin, license and version. Where content provenance standards are available, POCKET can expose them to the Carrier rather than implying all generated content is equally trustworthy.

23. Technical Threat Model

The project should assume both remote attackers and malicious ecosystem participants.

Threat	Example	Mitigation direction
Firmware compromise	Malicious update gains privileged access.	Secure boot, signed updates, rollback protection, staged rollout.
Model package tampering	Weights replaced with malicious artifact.	Content hashes, publisher signatures, reproducible manifests.
Prompt injection	Document instructs an agent to exfiltrate data.	Content/action separation, permission sandbox, confirmation gates.
Skill abuse	Third-party Skill accesses unrelated memory.	Capability permissions, sandboxing, network allowlists, review.
Wallet theft	Malware attempts to sign transaction.	Hardware-backed signing, user confirmation, transaction simulation.
Device theft	Attacker obtains physical unit.	Disk encryption, passcode/biometric gating, remote unlink, rate limits.
Sybil marketplace attack	Fake publishers manipulate ratings or grants.	Identity/reputation controls, deposits, review systems.
Burst data leakage	Remote provider receives unnecessary context.	Local redaction/minimization, provider policies, encrypted transit.
Offline double spend	Device spends same value with multiple peers.	Bounded vouchers/channels, later settlement, strict limits.
Supply-chain compromise	Malicious component or provisioning process.	Vendor controls, factory key ceremony, sampling, attestation audits.

23.1 Bug bounty and disclosure

Before public token or hardware launch, the project should publish a coordinated vulnerability disclosure policy, reward meaningful reports, and establish emergency procedures for firmware, contract and marketplace vulnerabilities. Security findings should be classified by exploitability and user impact rather than publicity value.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

24. Roadmap and Launch Gates

The roadmap is intentionally milestone-gated. Dates should be published only after the preceding gate is passed.

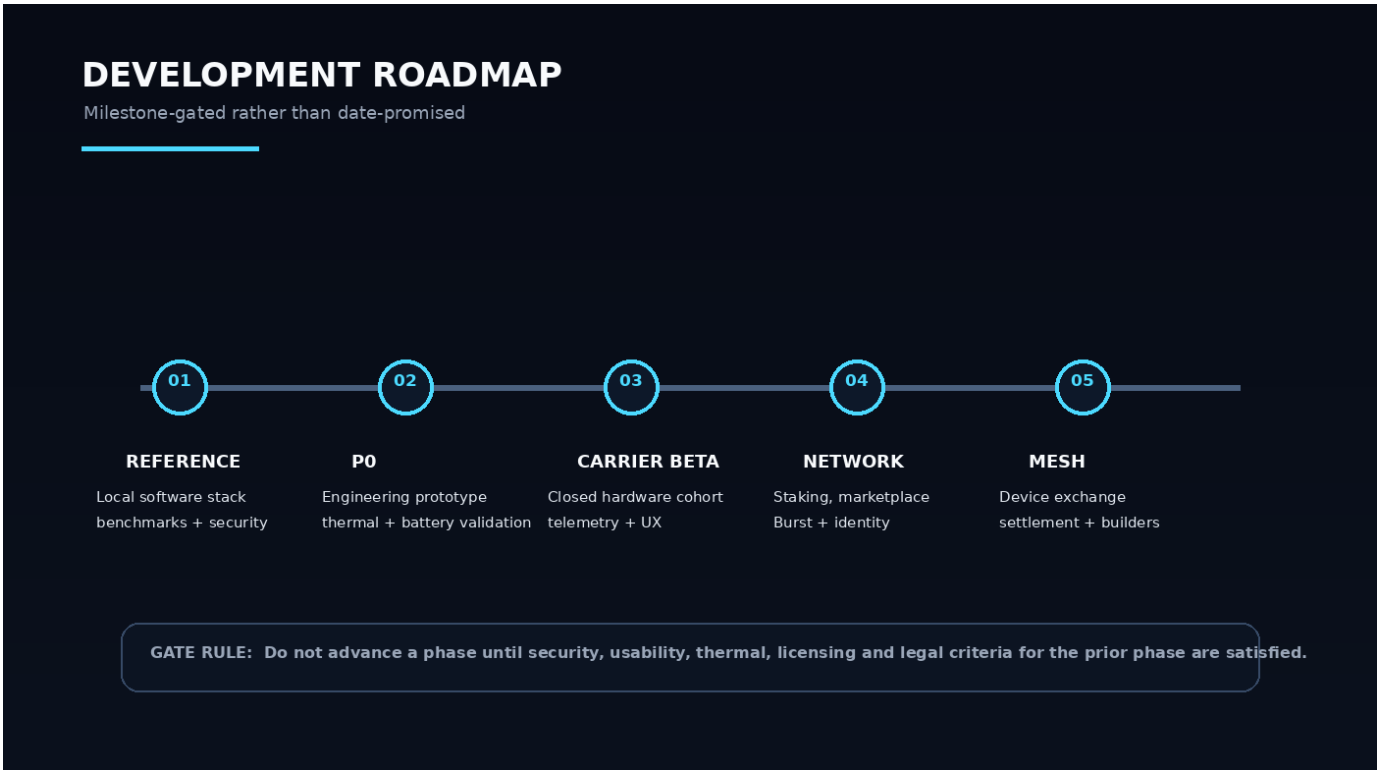


Figure 6. Proposed milestone sequence.

24.1 Phase 01 - Reference stack

- Select candidate edge hardware and inference runtime.
- Build local speech + language reference application.
- Benchmark latency, memory, thermals and battery on real target silicon.
- Define package signing, permission and secure-update architecture.
- Prototype wallet-to-device entitlement flow.

24.2 Phase 02 - P0 engineering prototype

- Produce working physical units.
- Validate microphones, speaker, camera, display and charging.
- Test sustained inference and thermal throttling.
- Implement secure device identity and manufacturing provisioning.
- Begin private usability studies.

24.3 Phase 03 - Carrier beta

- Small closed cohort with explicit telemetry consent.
- Measure daily usefulness, offline percentage, memory quality and failure modes.
- Harden update process and recovery tooling.
- Finalize first production Mind and Skill SDK.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- Publish hardware specifications only after measurement.

24.4 Phase 04 - Network launch

- Deploy audited $POCKET contracts.
- Launch access staking and entitlement service.
- Open curated marketplace.
- Introduce Burst compute and creator settlement.
- Publish supply, allocations, vesting and fee routing before token distribution.

24.5 Phase 05 - Mesh and open Builder ecosystem

- Device discovery and signed local exchange.
- Offline receipt experiments with conservative limits.
- Expanded Builder publishing and reputation systems.
- Additional hardware revisions and partner devices where justified.

25. Metrics and Network Health

The project should measure useful intelligence, not only token activity.

Category	Example metric	Why it matters
Product	Weekly active devices / retained Carriers	Shows whether the hardware solves recurring problems.
Local-first	Share of tasks completed without remote inference	Measures whether the core local thesis is real.
Quality	Task completion, correction rate, latency, battery cost	Balances intelligence against usability.
Memory	User-approved memory retention and deletion rates	Tests whether memory is valuable rather than intrusive.
Marketplace	Active Builders, paid installs, repeat purchases	Measures platform depth.
Creator economy	Creator earnings concentration and median payout	Detects whether value is broadly distributed.
Token utility	Share of token transactions tied to product/service actions	Separates utility from purely speculative transfer.
Staking	Distribution across tiers and unlock/unstake behavior	Tests whether tiers are accessible and valuable.
Security	Vulnerability rate, patch latency, malicious package detection	Measures operational trustworthiness.
Privacy	Percentage of tasks using Burst and average context transmitted	Quantifies data-minimization progress.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

26. Brand, Lore and Community System

The narrative makes POCKET memorable; the product makes the narrative credible.

26.1 Core line

POCKET - THE AI THAT ESCAPED THE CLOUD. The brand narrative expresses the local-first product thesis in a memorable way. It is lore, not a claim that a sentient model literally escaped a data center.

26.2 Lore sequence

Chapter	Narrative	Product idea beneath the lore
The Cloud	AI becomes enormous and centralized.	Cloud-first AI dominates the current experience.
The Compression	An experimental model reorganizes as it is shrunk.	Optimization and edge deployment.
The Escape	The model continues after the connection disappears.	Offline/local inference.
The Carriers	People carry individually owned intelligences.	Hardware ownership + user identity.
The Signal	Devices detect an unexplained transmission.	Future Mesh/network mystery and community storytelling.

26.3 Vocabulary

Term	Meaning
Carrier	POCKET holder/user in the ecosystem.
Mind	AI model/capability package.
Skill	Plugin, agent tool or integration.
Burst	Metered remote compute.
The Mesh	POCKET-to-POCKET network.
Builder	Developer/model creator.
P0	First engineering prototype.
The Signal	Ongoing mystery element in the lore.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

27. Key Risks and Mitigations

A credible whitepaper should state what can fail.

Risk	Potential impact	Mitigation direction
Hardware cost overrun	Device price becomes unattractive.	Multiple BOM scenarios, early supplier quotes, disciplined feature scope.
Battery/thermal limits	Local AI feels slow or drains battery.	Benchmark before industrial-design freeze; use tiered models and quantization.
Model quality gap	Local model underperforms user expectations.	Clear modes, specialist Minds, controlled Burst escalation.
Licensing restrictions	Desired model cannot be redistributed locally.	Prioritize permissive/commercially compatible licenses; negotiate rights where needed.
Token regulatory classification	Access or sale restricted in some jurisdictions.	Jurisdiction-specific counsel, utility-first rights, avoid equity/profit promises.
Token volatility	Service pricing becomes unpredictable.	Quote service prices in stable reference units and convert to $POCKET at execution if legally/technically appropriate.
Marketplace malware	Malicious Skills harm users.	Signing, review, sandboxing, permissions, revocation and security incentives.
Supply-chain delays	Hardware launch misses expectations.	Prototype early, dual-source where feasible, communicate milestone gates.
Founder entitlement cost	“Lifetime” benefit creates unsustainable service liability.	Define model family, local-access scope, fair-use and network-service exclusions precisely.
Speculative narrative overwhelms product	Community focuses on token price rather than utility.	Lead communication with working product, benchmarks, Builders and transparent development status.
Revenue-participation regulatory risk	Direct or indirect revenue-linked benefits may trigger securities, investment-product, tax, licensing or distribution restrictions.	Do not activate until jurisdiction-specific counsel approves the structure; geofence where required; publish exact rights, exclusions and tax treatment; avoid guaranteed-return marketing.
Revenue concentration / weak unit economics	An overly aggressive holder allocation could weaken hardware support, R&D or solvency.	Use net-revenue bases, reserves, source-specific caps, board/treasury controls and periodic rate review rather than

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

		promising a fixed percentage forever.

28. Legal and Regulatory Design Considerations

Final legal structure must be developed with qualified counsel in each launch jurisdiction.

28.1 Token rights

POCKET intends $POCKET to function as an access, settlement and participation token. If the Carrier Revenue Pool is activated, eligible token stakers may receive defined economic benefits funded from real ecosystem revenue. Those rights must be described exactly as implemented. They do not automatically constitute equity in POCKET Labs or ownership of company assets, but revenue-linked rights can still be regulated as securities, investment contracts, collective investment products, financial instruments or other regulated arrangements depending on jurisdiction and implementation.

28.2 Staking terminology

Because POCKET staking is proposed as both an access mechanism and, if activated, a qualification mechanism for revenue participation, documentation must clearly separate access benefits from financial participation. Reward sources must be disclosed. No fixed yield or APY should be implied. Where the revenue program is unavailable, staking may still function solely for model, service and ecosystem entitlements.

28.3 Consumer promises

Hardware specifications, battery life, model access and Founder entitlements should be published only when testable and contractually supportable. “Lifetime” language is particularly sensitive and should be accompanied by a complete definition, service scope, fair-use rules, compatibility limits and termination conditions.

28.4 Data protection

Local storage can reduce data exposure, but the project may still process personal information through accounts, payments, support, synchronization or Burst services. Privacy notices, consent flows, deletion procedures and cross-border processing arrangements should reflect the actual implementation rather than broad “your data never leaves the device” claims.

28.5 Revenue participation, securities and investment-product analysis

A token whose economic benefits are linked to product revenue requires materially more legal analysis than a pure access token. Before activation, qualified counsel should assess offering, secondary-market, staking, custody, money-transmission, consumer-protection, tax and financial-promotion rules in every supported jurisdiction. The project may need registration, exemptions, transfer restrictions, eligibility checks or a separate legal vehicle. The public website and whitepaper should never imply that compliance is solved merely by describing the token as utility.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

28.6 Jurisdiction gating and program activation

Revenue participation should launch only after a formal activation notice identifies the legal entity responsible for the program, eligible jurisdictions, excluded persons, revenue sources, calculation methodology, rates, distribution rail, reporting cadence, tax treatment, KYC/AML requirements if applicable and suspension/termination rights. Until that notice exists, all percentages in this paper are illustrative design targets only.

29. Conclusion

POCKET is an attempt to make personal AI physically and economically ownable.

The project combines dedicated hardware, a local-first AI runtime and a tokenized ecosystem. The hardware gives AI a persistent personal presence. The Mind gives the device intelligence. $POCKET coordinates access, model rights, creator economics, compute, network exchange and - if the compliant Carrier Revenue Pool is activated - participation in measurable revenue generated by the ecosystem itself. The design is deliberately structured so the product remains meaningful when disconnected while becoming substantially more capable and economically integrated when connected to the POCKET Network.

The strongest version of POCKET is not a token with hardware attached and not a gadget with tokenomics attached. It is a system in which each layer solves a problem the other layers cannot solve alone: local hardware provides privacy and presence; network services provide capability and scale; the token provides portable economic coordination across users, Builders, models, devices and compute providers; and real revenue, when legally activated for participation, creates a direct bridge between product adoption and long-term Carrier alignment.

Appendix A - Detailed Utility Matrix

Actor	Action	Uses $POCKET?	Result
Carrier	Stake to tier	Yes	Unlock tier-specific access and allowances.
Carrier	Install premium Mind	Yes	License/download signed local model package.
Carrier	Run Burst task	Yes	Receive remote inference/compute result.
Carrier	Buy Skill	Yes	Install Builder-created capability.
Carrier	Purchase knowledge pack	Yes	Store signed offline domain content.
Carrier	Request private tuning	Yes	Receive personalized adapter/model artifact.
Carrier	Register device	Potentially	Bind device identity to

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

			network entitlement.
Builder	Publish Skill	Potential bond/fee	List package in marketplace.
Builder	Earn usage fees	Receives $POCKET	Compensation tied to paid adoption.
Compute provider	Serve Burst job	Receives $POCKET	Compensation for measured compute.
Device	Settle Mesh receipt	Yes	Reconcile supported local service exchange.
Treasury	Fund grant	Pays $POCKET	Builder completes defined ecosystem deliverable.

Appendix B - Founder Program Design Checklist

- Name the exact flagship model family covered by the entitlement.
- Define whether entitlement requires continuous staking, a one-time qualification snapshot, a fixed lock term, or another rule.
- Define whether the entitlement is transferable, wallet-bound, device-bound or both.
- Define what “lifetime” means and the events that end the model family or program.
- State whether model updates inside the family are included.
- State whether third-party licensing costs are included or excluded.
- State Burst / cloud compute allowances separately from local model access.
- State device compatibility and minimum hardware requirements.
- State fair-use, abuse, sanctions and legal restrictions.
- Define recovery if the Carrier loses a device or wallet.
- Define whether a Founder may activate more than one device.
- Publish the terms before any user is asked to stake based on the benefit.

Appendix C - Example Skill Permission Manifest

A Skill should request only the capabilities it needs. Illustrative example:

```
{
  "skill_id": "builder.travel.assistant",
  "permissions": {
    "microphone": "session",
    "camera": "ask_each_time",
    "personal_memory": ["travel_preferences"],
    "network": ["api.example-travel.com"],
    "wallet": "no_signing"
  },
  "limits": {"storage_mb": 250, "background_minutes": 5}
```

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

```
}
```

Appendix D - Open Parameters Before Public Token Launch

Area	Decision required before launch
Legal entity	Issuer / operating entity, jurisdiction, terms and distribution restrictions.
Chain	Execution environment, wallet standards, fees, bridging policy and finality assumptions.
Supply	Total supply, decimals, genesis mint and non-inflation/inflation policy.
Allocation	Treasury, team, ecosystem, liquidity, partners and community distribution.
Vesting	Cliffs, linear vesting, lockups and on-chain transparency.
Staking	S1-S4 thresholds, lock periods, grace periods and Founder qualification.
Fees	Marketplace, Burst, creator share, treasury share and any burn mechanism.
Contracts	Independent audit, bug bounty, admin keys, timelocks and upgradeability.
Hardware	Final BOM, price, certifications, warranty, return policy and region support.
Model rights	Distribution licenses, commercial rights, attribution and update obligations.
Privacy	Data map, processor roles, retention, account recovery and sync terms.
Communications	Verified domain, social accounts, contact channels and anti-scam policy.

Appendix E - Glossary

Term	Definition
$POCKET	Proposed native access and settlement token of the POCKET Network.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Carrier	A user/holder participating in the POCKET ecosystem.
Mind	A signed local AI model or capability package.
Skill	A sandboxed integration, tool or agent extension.
Burst	Metered remote compute used when local capability is insufficient.
The Mesh	Local device-to-device communication and exchange layer.
Builder	Developer, model creator or other ecosystem publisher.
Founder	Highest proposed Carrier access tier.
Model Passport	Proposed entitlement credential for a defined model family.
P0	First engineering prototype milestone.
Secure element	Hardware-backed environment used to protect device identity or cryptographic keys.
Attestation	Cryptographic proof of selected device or software state.
Knowledge Pack	Curated, signed information package stored for local retrieval.
The Signal	Narrative/lore concept representing an unexplained transmission; not a technical protocol specification.

Appendix F - Whitepaper Change Log

Version	Date	Summary
0.9	30 Sep 2026	First comprehensive concept whitepaper integrating hardware, local AI, token utility, Carrier staking, Founder access, marketplace, Mesh, privacy, security and roadmap.
1.0	30 Sep 2026	Revenue Participation Edition: introduces proposed Carrier Revenue Pool, source-level product revenue allocations, time-weighted staking participation, Founder weighting, distribution rails, reporting standard and expanded legal/risk framework.
1.1	1 Oct 2026	Adds public reference

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

		hardware/software architecture, offline voice pipeline, factory provisioning model, productization gates, compliance-readiness framework and technical references.
1.2	1 October 2026	Added NVIDIA Jetson accelerator evaluation track, benchmark criteria, vendor-neutral shipping-silicon boundary and updated public technical references.

Next review should update all TBA parameters only after technical benchmarking, commercial modeling, smart-contract architecture and jurisdiction-specific legal review are complete.

Appendix G - Revenue Participation Reporting Standard

If the Carrier Revenue Pool is activated, each distribution period should publish a standardized report. The objective is to make the path from commercial activity to participant benefit reproducible and auditable.

Field	Required disclosure
Period	Start date, end date, reconciliation date and distribution date.
Eligible revenue by source	Hardware, subscriptions, marketplace, Burst, accessories, enterprise/OEM, partnerships and any additional approved category.
Source deductions	Taxes, refunds, chargebacks, processor fees, direct fulfillment, provider cost, warranty reserve and any other permitted deduction.
Eligible Net Revenue	Net amount for each source after published deductions.
Allocation rate	Percentage applied to each source for the period.
Pool contribution	Currency amount contributed by each source and aggregate Carrier Revenue Pool.
Eligibility rules	Stake threshold, minimum holding time, lock requirements, tier multipliers and concentration caps.
Total participation weight	Aggregate eligible weight used as denominator.
Distribution rail	Stable-value distribution, token-market action, service credit, hardware credit or other approved method.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Transaction evidence	Treasury transaction IDs, attestations or accounting references sufficient to verify the distribution.
Reserves	Opening reserve, additions, releases and closing reserve relevant to the program.
Jurisdiction limits	Regions or participant classes excluded from the period.

Appendix H - Public Technical Reference Architecture

This appendix translates the product thesis into a buildable public reference architecture. It is intentionally specific enough to show technical feasibility while remaining implementation-agnostic where supplier, security or manufacturing details are still under engineering review.

PUBLIC INFORMATION BOUNDARY. This whitepaper describes architecture, product goals and public reference components. It intentionally does not publish private keys, signing credentials, internal security procedures, unreleased source code, supplier pricing, factory test credentials, device provisioning secrets or other information that would reduce the security of the product or its users.

H.1 P0 reference hardware profile

The recommended first milestone is a functioning engineering prototype that proves the complete offline interaction loop before custom silicon or expensive tooling is committed. A compute-module-class architecture makes that possible while keeping the software path close to a future embedded product.

In parallel with the compute-module baseline, POCKET AI Labs can maintain an accelerator evaluation bench based on NVIDIA Jetson Orin Nano-class developer hardware. The objective is to compare local language, vision and multimodal workloads under a mature GPU toolchain while keeping the consumer handheld architecture vendor-neutral until power, thermal, cost and software measurements support a production choice.

Track	Public reference	Why evaluate it	Product status
Portable P0 baseline	64-bit Arm compute-module class	Prove the complete offline voice-to-answer experience on a compact, accessible prototype platform.	Reference prototype
Accelerator lab track	NVIDIA Jetson Orin Nano Super class	Benchmark GPU-accelerated LLM/VLM workloads, CUDA/TensorRT-class optimization and higher-throughput local AI.	Lab evaluation only
Shipping silicon	TBA after validation	Select only after latency, power, thermals, acoustic load, BOM,	Not selected

		supply and software portability are measured.	

Subsystem	Public reference profile	Engineering note
Compute	64-bit Arm compute-module class; Raspberry Pi Compute Module 5 is a practical P0 baseline	CM5 exposes a 2.4 GHz quad-core Arm Cortex-A76 SoC and configurations up to 16 GB RAM and 64 GB eMMC. Final production silicon is not locked.
Memory	16 GB-class for P0	Provides headroom for a small local LLM, speech services, OS and context cache.
Storage	64 GB eMMC baseline + optional fast expansion	Enough to prove preloaded models and the POCKET OS; production capacity should be determined from measured package sizes.
Display	3-4 inch touch display	Primary surfaces: state, short text, permissions, setup and the expressive POCKET face.
Audio	Dual/multi microphone + tuned speaker + physical action button	Voice-first without requiring always-listening wake-word behavior in the earliest prototype.
Vision	Compact camera / sensor module	Supports documents, objects and scene understanding on models and hardware that pass performance testing.
Connectivity	Wi-Fi + Bluetooth/BLE	Used for setup, updates, optional services and device pairing; core AI remains local-first.
Power	USB-C rechargeable battery system; early 4,000-5,000 mAh-class evaluation packs	Final battery claims follow measured workload, skin temperature and thermal throttling tests.
Identity	Hardware-backed device key storage where available	Supports signed device identity and offline entitlement verification without exposing raw secret material.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

H.2 POCKET OS and local runtime

POCKET OS is the device software image that binds the user interface, model runtime, memory, permissions, updates and device identity into one appliance. The factory image should boot into a usable local experience without requiring a model download.

Layer	Reference implementation direction	Offline behavior
Operating system	Hardened Linux-based embedded image	Boots and runs core services locally.
LLM runtime	llama.cpp-class GGUF inference runtime	No network required for installed models.
Language models	Qwen3-class 1.7B / 4B quantized models are public reference candidates	Fast and Deep modes can map to different model sizes or policies.
Speech recognition	whisper.cpp-class local ASR	Voice can be transcribed on-device.
Speech synthesis	Local ONNX/native TTS engine with distributable voice packs	Response audio is generated locally for supported voices.
Memory	Encrypted SQLite-class structured store + local vector/embedding index	User-approved memory remains on-device by default.
Skills	Signed, permission-scoped packages	Can execute locally when required resources are available.
Updater	Signed A/B or rollback-capable update mechanism	Network is used to fetch verified updates, not to authorize every inference.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

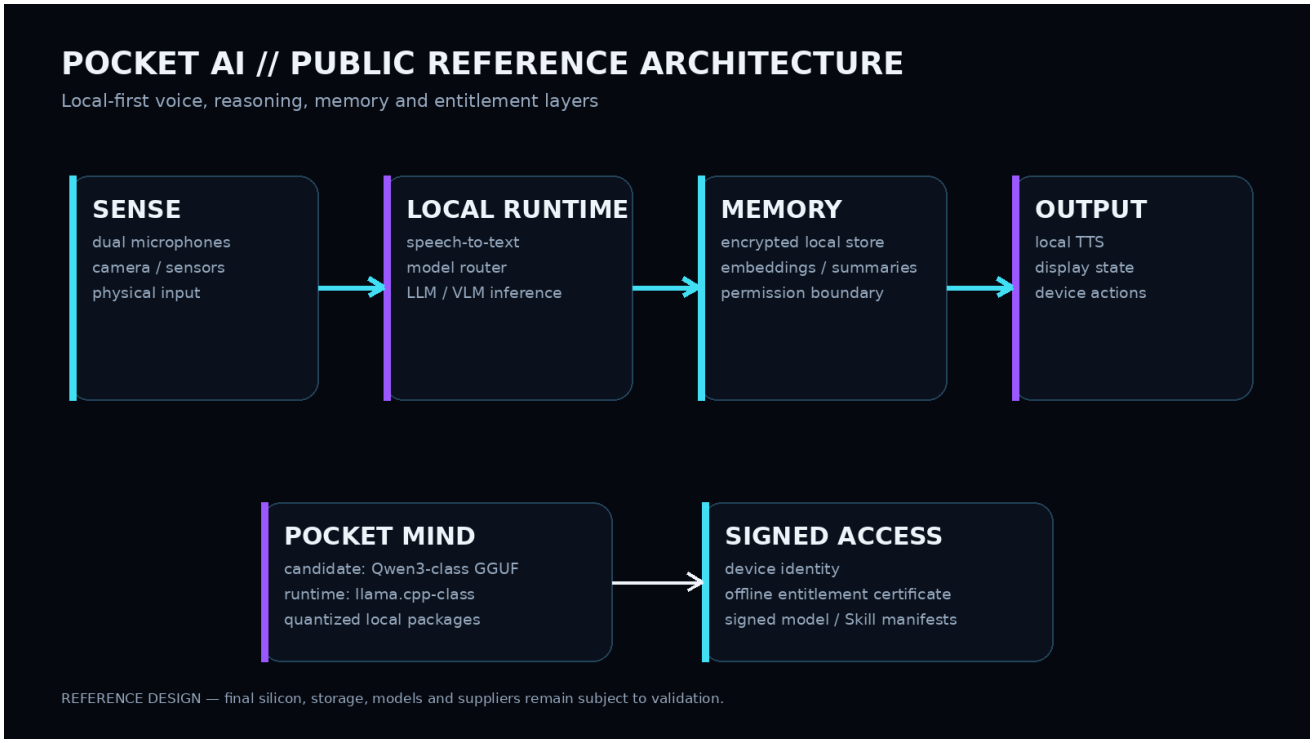


Figure H1. Public reference architecture for offline POCKET inference and signed access.

H.3 Offline voice-to-answer pipeline

The core interaction path is deliberately local. A button press or user-approved wake mechanism begins capture; speech is converted to text; the router selects the local Mind and permitted Skills; memory is read only under the active privacy policy; the answer is generated; and local TTS speaks the response. Connectivity is an escalation path, not the default dependency.

Stage	Function	Network dependency
1. Capture	Microphone records the active request	None
2. ASR	Local speech-to-text converts audio to text	None
3. Routing	Policy engine selects Fast, Deep, Skill or optional Burst	None for local paths
4. Context	Approved local memory and knowledge packs are retrieved	None
5. Inference	Installed Mind generates the response	None
6. Output	Local TTS + display render the answer	None
7. Optional escalation	Carrier-approved Burst or network service	Only when explicitly invoked or policy permits

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

H.4 Reference model strategy

The model layer should remain portable. The public reference stack can start with open-weight multilingual models that are small enough to quantify on the target hardware, while the POCKET Mind packaging format prevents the product from becoming permanently tied to one model family.

Qwen3 is a suitable reference family because its public release includes dense 1.7B and 4B variants, supports local use with llama.cpp, is multilingual, and is published under Apache 2.0. These facts make it useful for prototyping; they do not obligate the final commercial device to a single model provider.

H.5 Signed offline entitlement

POCKET token-gated services should not require a blockchain query for every prompt. When a Carrier qualifies for an access tier, the network can issue a signed entitlement to a registered device. The device verifies that entitlement locally and unlocks the corresponding installed model or service package while offline. Revalidation occurs according to published policy when the device reconnects.

Entitlement field	Purpose
device_id	Binds access to a registered POCKET device or approved device set.
tier	Identifies Carrier / Operator / Architect / Founder entitlement class.
capabilities	Lists Minds, Skills, compute allowances or service families unlocked.
issued_at / policy_epoch	Allows auditable policy updates without per-prompt network checks.
signature	Lets the device verify authenticity without exposing the issuer signing key.

H.6 Factory image and provisioning model

A production POCKET should leave the assembly line with a signed software image, validated local models and a unique device identity. The goal is an appliance experience: the customer powers on the device and the local Mind already exists.

Factory stage	Public description
Image flashing	Install signed POCKET OS, runtime components and approved base Mind package.
Device identity	Provision a unique hardware-backed identity or secure key container.
Calibration	Validate microphone, speaker, display, camera/sensors and battery telemetry.
Inference test	Run a deterministic local model test to confirm the installed AI stack launches and responds.
Update enrollment	Register the device for signed firmware/model updates without pre-authorizing user data upload.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Final QA	Record pass/fail manufacturing data and package only devices that meet the published quality gate.

H.7 Productization gates

POCKET AI Labs should move through staged engineering gates rather than attempting mass production directly from renders. Each phase reduces a different class of risk.

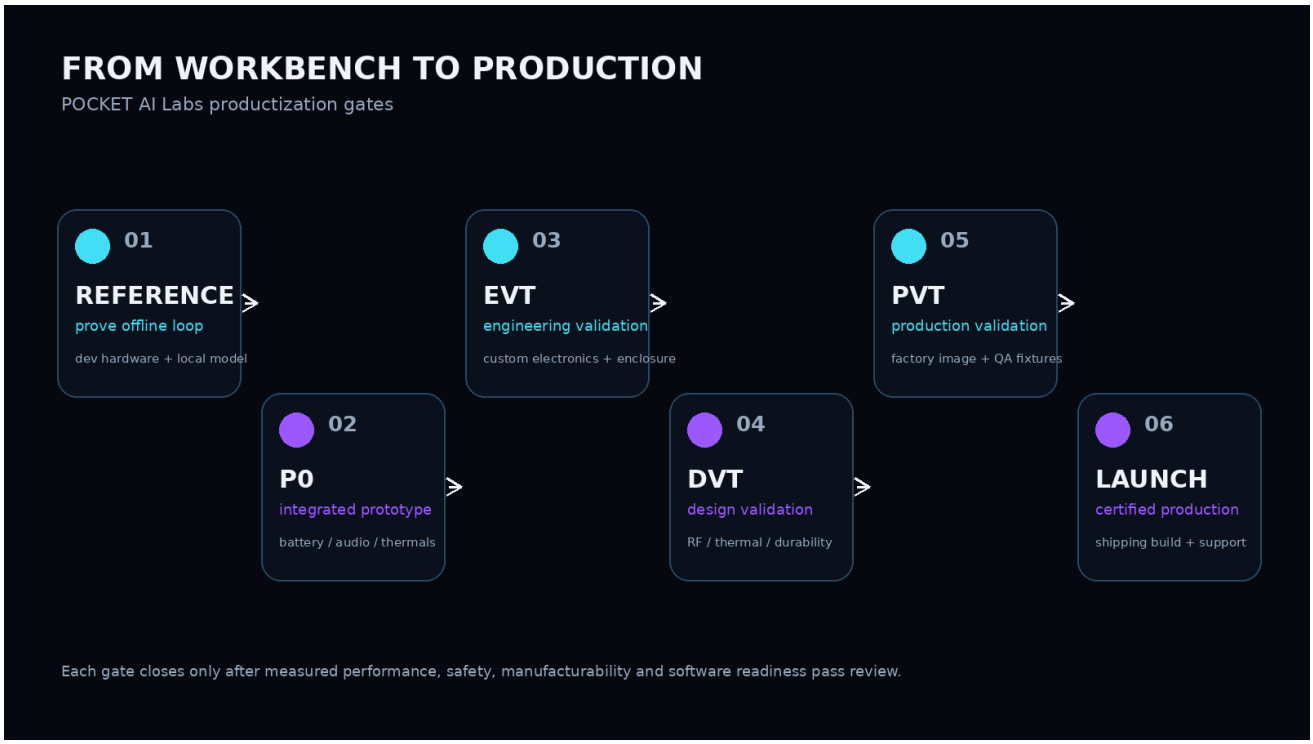


Figure H2. Reference productization gates from local prototype through certified production.

Gate	Primary proof
Reference stack	Offline speech -> model -> local speech loop works on development hardware.
P0	Integrated handheld prototype validates latency, thermal behavior, battery draw and interaction model.
EVT	Custom electronics and mechanical package validate electrical design, interfaces and manufacturability.
DVT	Near-final enclosure, RF, thermal, acoustics, durability and software update path are validated.
PVT	Production line, factory flashing, calibration, QA fixtures and packaging are validated at pilot quantity.
Launch	Only the certified, tested configuration becomes the shipping specification.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

H.8 Compliance and release readiness

The final certification matrix depends on the exact radio modules, battery, power architecture, markets and enclosure. A consumer wireless device should be designed from the beginning for the applicable radio, electromagnetic compatibility, safety, battery transport, privacy and cybersecurity requirements in target markets. For the European Union, the Radio Equipment Directive establishes essential requirements for radio equipment, including safety, electromagnetic compatibility and efficient radio-spectrum use; final obligations must be confirmed against the shipping configuration.

H.9 Engineering principles for public claims

Principle	Public rule
Measured, not imagined	Battery life, speed, thermal performance and supported model sizes are published only after repeatable hardware testing.
Reference, not lock-in	Named open-source components describe a feasible baseline, not a permanent supplier commitment.
Local means local	Features advertised as offline must remain functional after network removal once legitimately provisioned.
Security by omission	Public documents explain trust boundaries without publishing secrets that could weaken device security.
Upgradeable architecture	Model packages declare runtime, memory, modality and license requirements so future hardware generations remain compatible where possible.

H.10 NVIDIA Jetson Edge AI accelerator evaluation track



POCKET AI Labs accelerator evaluation-track graphic. This is not an NVIDIA corporate logo and does not represent partnership or endorsement.

NVIDIA Jetson Orin Nano Super is a useful public benchmark platform for POCKET AI Labs because NVIDIA documents up to 67 INT8 TOPS of AI performance, an Ampere GPU with 1,024 CUDA cores and 32 Tensor Cores, 8 GB of 128-bit LPDDR5 memory with 102 GB/s bandwidth, and configurable module power between 7 W and 25 W. Those characteristics make it valuable for laboratory work on local LLMs, vision-language models, multimodal routing and optimized inference. They also illustrate why an accelerator

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

platform must be evaluated against handheld constraints rather than assumed to be the production answer.

POCKET AI Labs can use this evaluation track to compare llama.cpp-class portability against GPU-optimized paths such as CUDA and TensorRT-LLM, measure time-to-first-token and sustained token throughput, evaluate multimodal models, profile memory pressure, and quantify the thermal and acoustic implications of accelerated edge inference. A successful benchmark on Jetson would validate software and model possibilities; it would not by itself justify using the same module in the final pocket-sized product.

Evaluation metric	What POCKET AI Labs measures	Decision implication
Latency	Time-to-first-token, tokens/sec, speech round-trip and vision response time.	Determines whether acceleration materially improves the user experience.
Power	Idle, active inference, peak draw and energy per completed task.	Shows whether the platform fits the battery target.
Thermals	Sustained enclosure temperature, throttling and cooling requirement.	Determines whether silent handheld packaging is realistic.
Memory	Model load footprint, KV-cache pressure and multimodal memory usage.	Defines practical local model sizes and context windows.
Toolchain	Portability, CUDA/TensorRT-LLM gains, update complexity and fallback paths.	Balances performance against software lock-in and maintenance cost.
Supply / BOM	Module availability, lifecycle, integration cost and production alternatives.	Prevents a benchmark platform from becoming an unexamined production dependency.

Brand and affiliation boundary. NVIDIA and Jetson are referenced in this paper only as third-party technologies that may be evaluated by POCKET AI Labs. POCKET AI Labs is not claiming a partnership, sponsorship, endorsement or supplier commitment. Any future use of NVIDIA corporate logos or partner co-branding would require the permissions and brand usage rights applicable at that time.

Appendix I - Public Technical References

The following public references support the feasibility statements in Appendix H. They are reference technologies, not endorsements, partnerships or final production commitments.

NVIDIA Jetson Orin Nano Super Developer Kit — NVIDIA lists 67 INT8 TOPS, 1,024 CUDA cores, 32 Tensor Cores, 8 GB 128-bit LPDDR5 at 102 GB/s and 7 W-25 W module power. NVIDIA also documents support for local AI stacks including llama.cpp and TensorRT-LLM. Referenced as a laboratory evaluation platform, not a supplier or production commitment.
https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/
https://developer.nvidia.com/embedded/jetson-modules

Raspberry Pi Compute Module 5 — Official specifications include a 2.4 GHz quad-core 64-bit Arm Cortex-A76 SoC, up to 16 GB LPDDR4 memory, up to 64 GB eMMC, Wi-Fi and Bluetooth options, and a 55 x 40 mm module footprint.
https://www.raspberrypi.com/products/compute-module-5/

Qwen3 — Public Qwen3 release documentation lists 1.7B and 4B dense variants, 100+ languages/dialects, local llama.cpp workflows, and Apache 2.0 licensing for open-weight models.
https://github.com/QwenLM/Qwen3

llama.cpp — Open-source C/C++ LLM inference runtime distributed under the MIT License.
https://github.com/ggml-org/llama.cpp

whisper.cpp — Open-source local speech-recognition implementation supporting Raspberry Pi and fully offline use cases; MIT License.
https://github.com/ggml-org/whisper.cpp

EU Radio Equipment Directive — European Commission overview of the regulatory framework for placing radio equipment on the EU market.
https://single-market-economy.ec.europa.eu/sectors/electrical-and-electronic-engineering-industries-eei/radio-equipment-directive-red_en

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Appendix J - POCKET Mind In-House Intelligence Program

POCKET Mind is the intelligence product layer developed by POCKET AI Labs for the browser preview and future POCKET hardware. It is designed as a model family plus runtime, memory, voice, tool-use and safety system rather than as a single interchangeable checkpoint.

J.1 What POCKET Mind means

POCKET Mind is the in-house intelligence stack that defines how a POCKET behaves. POCKET AI Labs owns and develops the interaction architecture, model routing, system behavior, memory rules, permissions, tool interfaces, local deployment packaging, evaluation harness and product personality. Early public builds may use suitable open-weight foundation models beneath this stack while POCKET-specific fine-tuned and distilled checkpoints are developed. The project should not describe an open-weight foundation checkpoint as trained from scratch by POCKET AI Labs unless that is factually true.

- POCKET Mind Web - an instant cloud-connected browser companion used for public onboarding, general-purpose conversation and product demonstration.
- POCKET Mind Edge - the device-oriented language model profile optimized for local inference, low memory use and predictable latency.
- POCKET Voice - the speech layer for local speech recognition, turn detection and speech synthesis.
- POCKET Memory - user-controlled persistent memory, preference storage and retrieval policy.
- POCKET Skills - the permissioned tool and agent interface used for notes, translation, device controls, local files and future third-party extensions.
- POCKET Router - the policy layer that chooses the appropriate local Mind, Skill or optional Burst path for each request.

J.2 Web preview as the first public POCKET Mind

The website functions as the first public POCKET Mind surface. A visitor can open the site and interact immediately with a persistent animated POCKET companion by text or, after granting browser microphone permission, by voice. The visitor does not download model weights. The web interface connects to a server-side POCKET Mind inference endpoint while preserving the same POCKET personality, state language, memory concepts and Skills architecture intended for the physical device.

POCKET Mind Web is intentionally separated from the edge-device runtime. The web experience uses server-side inference so onboarding is immediate and can be backed by a high-capability general-purpose model. The physical POCKET roadmap remains local-first: validated device models, speech recognition, speech synthesis, memory and core Skills are intended to continue operating locally, with optional network inference used only when additional capability is desired. Browser microphone and speech output remain subject to browser permissions and platform support.

J.3 Device model program

The POCKET device model program should progress through measurable model gates rather than branding one checkpoint as final. The target is a multilingual, instruction-following model that remains useful inside a tight memory, thermal and battery envelope.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- Foundation selection - benchmark multiple commercially usable open-weight base models for quality, multilingual coverage, memory use and license compatibility.
- POCKET instruction tuning - fine-tune on a POCKET-authored instruction set covering concise assistance, local-device behavior, privacy-aware responses, tool invocation and recovery from uncertainty.
- Distillation and quantization - compress the model into device-appropriate variants while preserving instruction following and safety behavior.
- Tool-use training - teach structured invocation of POCKET Skills instead of forcing the language model to answer every task from parametric memory.
- Evaluation - maintain repeatable suites for latency, hallucination rate, multilingual quality, tool accuracy, memory behavior, battery impact and safety.
- Signed release packaging - ship only validated model artifacts through POCKET OS update channels with versioned manifests and rollback support.

J.4 Proposed model family

Profile	Primary surface	Design priority	Status
POCKET Mind Web	Website onboarding	Fast download, browser compatibility, clear product personality	Public preview architecture
POCKET Mind Fast	Physical POCKET	Low latency, everyday commands, translation, notes and routing	Engineering target
POCKET Mind Deep	Physical POCKET	Higher reasoning quality for harder local tasks	Engineering target
POCKET Mind Pro	Higher-memory future hardware / optional compute	Larger context, specialist tools and premium capabilities	Future profile

J.5 Voice interaction architecture

A natural POCKET interaction is a pipeline, not only a language model: microphone capture -> voice activity detection -> speech recognition -> POCKET Router -> POCKET Mind -> memory and Skills -> response text -> speech synthesis -> animated device state. Each stage should be independently replaceable so that speech, language and tool components can improve without redesigning the entire product.

J.6 Memory and personalization

POCKET Memory should remain explicit and inspectable. The assistant may maintain local preferences, named facts, recent conversational context and Skill-specific state, but the user should be able to review, edit, export or clear persistent memory. Sensitive categories should require stricter consent and retention rules. Model personalization should prefer retrieval and user-controlled memory before any irreversible model fine-tuning on personal data.

J.7 Safety and truthfulness requirements

- The assistant should distinguish local device state, retrieved facts and model-generated reasoning.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- The interface should make uncertainty visible instead of fabricating device capabilities or network status.
- The browser preview must disclose when voice capture or speech output relies on browser services rather than local POCKET code.
- Public materials should distinguish POCKET AI Labs-developed layers from third-party foundation models and runtimes.
- Model updates should be versioned, signed and reversible during engineering validation.

J.8 Web-to-device continuity

The purpose of the browser POCKET Mind is not to create a separate chatbot product. It is an onboarding surface for the same interaction model planned for the hardware. The website and device should share the same naming, state machine, personality specification, Skill schema and memory concepts so that a user who tests POCKET on the website already understands how the physical device behaves.

J.9 Development status boundary

POCKET Mind should be described as an active in-house AI software program and product architecture. Until POCKET AI Labs has trained, evaluated and released its own fine-tuned checkpoints, public materials should avoid implying that every underlying model weight was created from scratch internally. This distinction does not reduce the value of the in-house system: modern AI products commonly combine foundation models with proprietary data, orchestration, safety, memory, tools, inference optimization and product-specific fine-tuning.

J.10 Public website implementation

The POCKET website includes a live POCKET Mind companion that follows the visitor through the site, can be repositioned by the user, reacts visually to listening, thinking and speaking states, and connects to a server-side general-purpose language model through the POCKET Mind API. Its role is broader than a scripted project guide: it can hold normal conversations and assist with general questions while retaining deep project-specific context when the user asks about POCKET. The companion stores only the conversational context explicitly configured by the product and should provide visible controls for resetting or managing memory.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Appendix K — 2027–2030 Strategic Scenario Roadmap

POCKET AI Labs uses a long-range scenario roadmap to connect engineering, product, distribution, POCKET Mind, network infrastructure and commercial scale. The figures in this appendix are planning scenarios rather than forecasts, promises, investment guidance or guaranteed outcomes.

K.1 Roadmap design principles

- Engineering before scale — device reliability, local AI performance, thermal behavior, battery life and factory QA must be proven before aggressive production expansion.
- POCKET Mind as a product layer — web, edge and hybrid Minds should share behavior, memory concepts, Skills and identity while allowing different model backends and compute envelopes.
- Revenue diversification — hardware sales should be complemented by recurring AI services, marketplace activity, accessories, enterprise/OEM licensing and compute where commercially viable.
- Exchange access is conditional — centralized-exchange availability depends on legal eligibility, exchange diligence, custody support, liquidity, market integrity and third-party approval. No exchange listing is assumed or guaranteed.
- Scenario ranges, not guidance — unit and revenue ranges are internal planning cases designed to test operational requirements and capital needs.

K.2 Illustrative 2027–2030 scenario

Year	Primary objective	Illustrative device scale	Illustrative revenue range	Network / POCKET Mind	Centralized-exchange objective
2027	Prove + launch	Pilot / early production	$2M–$8M	P0/P1, POCKET Mind Web, offline voice stack, Skills beta	CEX readiness, legal/custody/liquidity work and eligible listing applications; no listing assumed.
2028	Scale the device	25K–60K units	$20M–$60M	POCKET Mind Web + Edge + Hybrid, multilingual voice, marketplace expansion	Broader global CEX applications where legally eligible and technically supported.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

Year	Primary objective	Illustrative device scale	Illustrative revenue range	Network / POCKET Mind	Centralized-exchange objective
2029	Build the platform	100K–250K ecosystem units	$90M–$250M	Mesh services, authenticated devices, creator distribution, enterprise/OEM expansion	Objective of deeper regulated liquidity and institutional-grade custody, subject to approval.
2030	Global personal-AI layer	350K–750K+ annualized ecosystem scale	$300M–$750M	Mature device/model/Skills flywheel, expanded form factors and global Carrier infrastructure	Broad market-access objective across eligible jurisdictions; never treated as a guaranteed outcome.

K.3 2027 — prove the complete loop

The 2027 scenario is centered on proving that POCKET works as a complete product rather than optimizing for headline scale. The core milestone is a device that can boot into POCKET OS, accept voice input, execute a local or hybrid POCKET Mind request, speak a response, maintain controlled memory, verify device identity and receive signed software/model updates.

- Complete P0, EVT and DVT hardware gates and validate microphones, speakers, display, power, thermal behavior, enclosure and production test fixtures.
- Operate POCKET Mind Web as the public onboarding surface while validating POCKET Mind Edge on device-class hardware.
- Launch an early Skills framework and developer documentation so useful device actions are not limited to free-form language generation.
- Establish manufacturing, certification, warranty, support and secure factory provisioning processes.
- Prepare token infrastructure, custody compatibility, market-integrity controls and exchange application materials only where legally eligible.

K.4 2028 — scale product and recurring software

The 2028 scenario assumes a transition from engineering-led production to a repeatable consumer hardware program. The commercial model begins shifting from primarily device sales toward a blend of hardware, recurring AI services, marketplace economics and enterprise pilots.

- Expand POCKET Mind into clearly versioned Web, Edge and Hybrid variants with shared identity, memory and Skills conventions.
- Introduce multilingual voice, improved local reasoning, signed model packages and a broader catalog of specialist Minds.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.

- Scale production only after return rates, field reliability, support load and unit economics meet published internal gates.
- Pursue broader exchange access through formal applications rather than marketing listings before approval.

K.5 2029 — platform economics

The 2029 scenario treats POCKET as a platform rather than a single device. The economic goal is to increase the share of recurring and ecosystem revenue while expanding device distribution and enterprise/OEM relationships.

- Operate a mature Skills and model marketplace with creator publishing, testing, signing, discovery and settlement.
- Expand authenticated device-to-device services and secure entitlement portability across user-owned POCKET hardware.
- Develop enterprise and OEM licensing paths for POCKET Mind components, device software and specialized hardware variants.
- Deepen custody, liquidity and exchange infrastructure only in markets where the token structure is legally supportable.

K.6 2030 — global personal-AI infrastructure

The 2030 scenario imagines POCKET as a distributed personal-AI layer spanning dedicated handhelds, accessories, specialized form factors, web surfaces and embedded/OEM deployments. The objective is a durable flywheel between POCKET AI Labs research, device adoption, recurring software, developer activity and network utility.

K.7 Revenue scenario methodology

The revenue ranges above are illustrative scenario bands, not management guidance. They are intended to test operating requirements under combinations of device average selling price, production volume, accessory attach, recurring AI-service conversion, marketplace take rate, enterprise contracts and compute usage. Actual results could be materially lower, higher or nonexistent depending on product-market fit, execution, financing, supply chain conditions, regulation and competition.

- Hardware: device and developer-kit sales net of returns, discounts, taxes and channel deductions.
- AI services: premium model access, storage/synchronization, Burst compute and other paid services where offered.
- Marketplace: platform fees associated with third-party Minds, Skills or content, subject to creator economics and applicable law.
- Enterprise/OEM: software, device or model licensing and integration programs.
- Accessories and services: approved peripherals, support programs and related commercial offerings.

K.8 Centralized-exchange pathway

A centralized-exchange roadmap must be treated as a readiness and application process rather than a listing promise. Exchanges independently determine whether to list an asset and may require legal opinions, market surveillance, liquidity support, custody integration, technical reviews, token-distribution information, issuer/project diligence and jurisdiction-specific restrictions. POCKET should never represent an exchange as confirmed until the exchange itself has approved and publicly announced the listing.

NVIDIA / CUDA / TensorRT / Jetson are third-party technology references used for independent evaluation. No affiliation or endorsement implied.