ChatGPT

Blueprint for an Autonomous Open-Source AI Platform

Introduction

Building a **"billion-dollar" AI platform** requires combining cutting-edge technologies with a sustainable, cost-effective strategy. We envision an open-source environment where AI agents can **grow, learn, and generate revenue** autonomously. This means leveraging the **newest free and open-source software** – from advanced language models to agent frameworks – to create a system that *actually works* (not just spits out content). By integrating these tools and hosting on free platforms (e.g. Next.js on Netlify), we can minimize costs while maximizing innovation. Below is a deep-dive into the technologies and steps needed to accomplish this vision. *(We will use Markdown headings, bullet points, and source citations throughout for clarity.)*

The Latest Open-Source AI Tech (Our AI "Brain")

A powerful AI system starts with powerful models. Recent advances have produced **open-source large language models (LLMs)** that rival or even surpass proprietary systems, providing the "brain" of our platform:

- **Meta's LLaMA 4:** Meta continues its open-source leadership with **LLaMA 4**, a state-of-the-art multimodal model (e.g. variants like *Llama 4 Scout* and *Llama 4 Maverick* process text, images, etc.) [1]. LLaMA 4 Scout boasts an industry-leading **10 million token context window**, enabling analysis of massive documents in one go [2]. Unlike closed models from OpenAI/Google, LLaMA's open release lets us fine-tune and deploy it on our own infrastructure – giving flexibility and control [3]. Impressively, LLaMA 4's performance in coding, reasoning, and multilingual tasks **outperforms** many competitors (it even beats GPT-4-level systems on some benchmarks) [4].
- **Alibaba's Qwen-3 Series:** Alibaba has open-sourced its latest **Qwen 3** LLM family under an Apache 2.0 license [5]. These models range from 4B up to a massive 235B parameters, using a Mixture-of-Experts (MoE) architecture for efficiency [6] [5]. Qwen-3 models reportedly **match or exceed GPT-4** on many benchmarks [7], and specialized variants like *Qwen-3-Coder* (for coding), *Qwen-VL* (vision+language), and *Qwen-Audio* (speech) are available for targeted tasks [5]. This gives us a whole toolbox of high-performance models we can freely use and fine-tune.
- **DeepSeek V3.1:** An emerging open model from China, **DeepSeek V3.1** (released August 2025) offers an all-in-one solution for chat, coding, and logical reasoning [8]. It's open-sourced under the **MIT license** (meaning free for commercial use and modification) [9]. DeepSeek V3.1 uses a hybrid "thinking" vs "fast" mode to balance complex reasoning with speed, and it employs an MoE architecture with multi-head attention to handle context windows up to **128k tokens** [10]. In practice, many organizations have adopted DeepSeek V3 as a go-to for large-context tasks like analyzing lengthy financial reports or codebases [8].
- **Mistral AI Models:** Startup Mistral AI provides a diverse **portfolio of open models**. Their premier models (like *Mistral Medium 3* and *Magistral*) are API-only for enterprises, but importantly Mistral also

releases **fully open models under Apache-2.0** [11]. For example, *Mixtral-8×22B* (an 8-expert 22B-parameter MoE model) delivers high performance efficiently [11]. They also offer **smaller "edge" models** (Ministral 3B, 8B) designed for resource-constrained devices [12] – perfect for running AI on local hardware or free-tier servers. Additionally, specialized open models like *Devstral* (for code generation) and *Pixtral* (for image tasks) indicate we can pick the right tool for each job [11].

- **Open Source from OpenAI:** Even OpenAI has begun contributing to open source. Notably, they released **GPT-oss-120B and GPT-oss-20B**, "open-weight" models under Apache 2.0 [13]. These smaller GPT variants offer strong performance and are optimized for efficient deployment – reportedly **able to run on consumer hardware** while excelling at "agentic" tool-using workflows and function calling [13]. This trend shows that we can obtain top-tier AI capabilities **without paying for API access**, by using open implementations of advanced architectures.

Why does this matter? Using these state-of-the-art open models as our AI core means we have no licensing fees or pay-per-call costs, and we can customize them freely. We can also **mix and match models** for different tasks (e.g. a large LLaMA-4 for general reasoning, plus a smaller Qwen-3 variant for specialized coding or a Mistral model for on-device tasks). These models' huge context windows and multimodal abilities open up possibilities to handle complex real-world data (images, long texts, etc.) within a single system. And critically, being open-source, they can be **fine-tuned or improved continuously** as we gather more data – which leads to the next point.

AI Agents and Autonomy (Beyond Content Generation)

Having a strong model is not enough; we need a system for the AI to **take actions, use tools, and perform multi-step tasks autonomously**. In other words, we must wrap the model in an **"AI agent" framework** that allows it to **interact with its environment** (e.g. browse the web, write to files, execute code) and **pursue goals with minimal human input**. Fortunately, 2024–2025 saw an explosion of open-source frameworks for exactly this purpose. Below are some of the top frameworks we can leverage (and even combine) to give our AI agent robust capabilities:

- **LangChain:** A popular toolkit for building LLM-powered applications [14]. LangChain makes it easy to chain model prompts together, integrate external tools, and manage conversational **memory**. It's a foundational library we can use to orchestrate prompts and tool usage in our system (for example, connecting the AI to a web search or a calculator when needed).
- **Auto-GPT:** The project that sparked mainstream interest in autonomous agents [15]. **Auto-GPT** (open-source on GitHub) showed that by looping GPT's outputs back into itself, the AI can recursively **plan and execute tasks** toward a goal with very little human intervention. It provides features like long-term memory, internet browsing, and file system access [15]. This means our system can, say, receive a high-level objective ("launch an e-commerce site") and then **iteratively break it down**, search for information, generate content/code, test results, and adjust – all on its own.
- **MetaGPT:** An agent framework that simulates a **team of AIs collaborating** to build software [16]. MetaGPT assigns different "roles" to agents (e.g. CEO, Project Manager, Developer) and lets them communicate to plan and develop a product autonomously [16]. This is perfect for complex projects – our AI could effectively become a **virtual startup team**, brainstorming features, writing code, and deploying an app or service with minimal oversight. It's built on open models and can drastically speed up product development cycles.
- **CAMEL (Camel-AI):** A novel framework where two or more AI agents **role-play conversations to solve tasks** [17]. For instance, one agent can play the "user" and another the "assistant" as they work

through a problem together [17]. This approach has interesting implications for training and alignment – by letting AIs debate/clarify with each other, they can reach solutions that a single-agent might miss. We could use CAMEL to allow our AI to **self-refine its plans** (one agent critiquing or double-checking another's approach).

- **SuperAGI:** A full-stack **"AGI" orchestration platform** with a graphical interface [18]. SuperAGI supports integrating multiple tools and APIs, has built-in memory (including vector database integration for long-term knowledge), and is designed for **enterprise-scale agent deployments** [18]. We can use it to manage our AI's various skills in a modular way. For example, SuperAGI could coordinate one sub-agent that handles marketing, another that handles research, etc., all sharing a memory store. Its support for vector DBs means we can plug in a knowledge base that our agents query as needed (more on that soon).
- **Open Agents (formerly OpenDevin):** An open-source framework focused on **AI-assisted software development** [19]. Open Agents provides an environment where an agent can plan, write, and **execute code in a sandbox** (with safeguards) [19]. This is hugely beneficial for a self-improving system – our AI could literally rewrite or extend parts of its own codebase, test the changes, and deploy updates. Open Agents' design as a developer tool means we could trust the AI to handle certain programming or debugging tasks automatically, effectively **iterating on itself** to get better over time.

(Note: There are many more frameworks – e.g. BabyAGI for simple task loops, Microsoft's AutoGen for multi-agent chats, CrewAI for role-based agent teams, etc. – but the ones above are the most relevant to our goals. We can mix elements from these or even run multiple in parallel for different functions.)

By combining these agent capabilities, our platform will go **far beyond basic content generation**. The AI will be able to: search and scrape information online, use APIs/tools (for example, to trade stocks or send emails), divide complex goals into subtasks, and collaborate with other AI "specialists." In short, we're giving the AI an **"action layer"** on top of its raw intelligence. OpenAI's own research emphasizes this direction – they optimized their open GPT-oss models specifically for **agentic workflows and tool use** [13], highlighting that an AI which can act is much more powerful than one that only chats.

Continuous Learning and Self-Improvement

To truly "grow and learn" autonomously, the AI system should improve itself over time. This involves two things: **learning from new data/experience** (so performance gets better the more it works), and **adapting to changing goals or environments**. We can achieve continuous learning through several open-source approaches:

- **Reinforcement Learning from Feedback:** We can incorporate human feedback or defined success metrics to fine-tune the models in an ongoing loop. Importantly, there are now open-source pipelines to do full-scale **RLHF (Reinforcement Learning with Human Feedback)** training on our own data. For example, the **OpenRLHF** project provides a high-performance framework (built on Ray and Hugging Face Transformers) that can train models up to 70B parameters with distributed optimization [20]. Using such tools, we could periodically retrain or refine our AI on the feedback it gets – e.g. if users respond that certain answers were poor, we can have the AI learn to avoid those in future. This continuous fine-tuning cycle means our AI *doesn't stay static*: it gets better at its job as time goes on, much like an employee gaining experience.

- **Automated Self-Reflection:** Some agent frameworks implement a loop where the AI reviews its own outputs and adjusts. A great example is **BabyAGI's** simple but effective approach – it generates tasks, prioritizes them, executes them, then **uses the results to inform the next tasks** in a feedback loop [21]. In practice, this might mean our AI generates a plan, attempts it, and if it fails or results are suboptimal, it learns from that outcome and revises its strategy. We will design our system such that the AI keeps logs of its actions and outcomes (a "memory") and periodically analyzes them to identify mistakes or new opportunities. Over time, this results in an agent that **learns from its own history**.

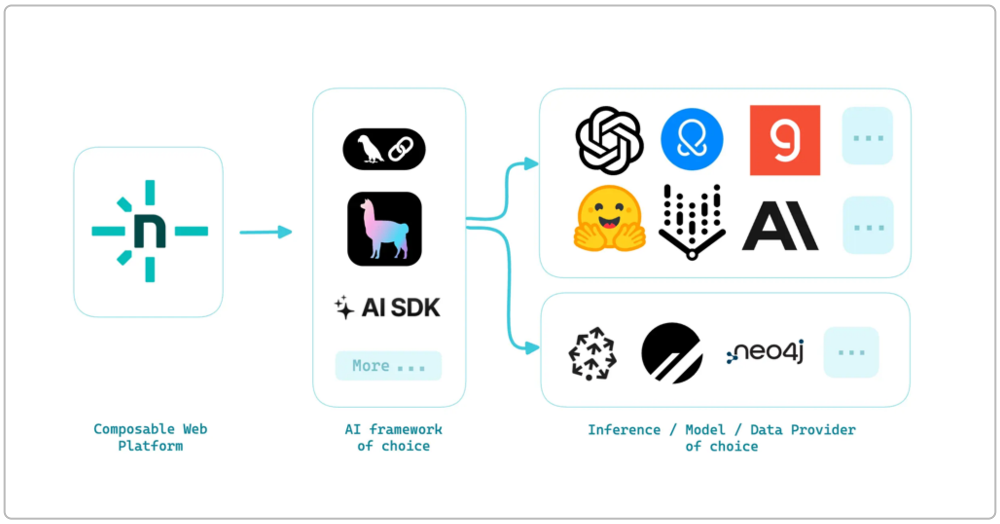
- **Multi-Agent Learning and Evolution:** Since our platform may involve multiple specialized AIs (as described with MetaGPT, CAMEL, etc.), we can also employ competitive or cooperative learning. For instance, we could have two agents with the same goal but different methods compete to see which performs better, effectively using **self-play** to discover best strategies. There's even work towards training multi-agent systems via reinforcement learning – e.g. a recent extension of OpenRLHF called **MARTI** is designed to train LLM-based multi-agent setups using RL with centralized coordination [22]. This means in the near future, we could directly train a team of AIs to optimize their collaboration. While this is cutting-edge, our platform could be built with hooks to incorporate such techniques as they mature (leveraging open research code).

- **Knowledge Base Expansion:** As the AI operates, it will accumulate data (customer queries, market info, code libraries, etc.). We'll use an open-source **vector database** (like Chroma or Weaviate) to store embeddings of important information, enabling the AI to recall and **build its own knowledge repository**. This is akin to the AI "learning" new facts on its own. Many agent frameworks (e.g. SuperAGI) support vector DB integration for long-term memory [18], and Netlify's platform can connect to services like Pinecone or Supabase for this purpose [23]. By continuously updating this knowledge base and retraining on it when needed, the AI stays up-to-date. Imagine the AI reading daily financial news and adding key insights to its memory store – it would gradually become an ever more savvy financial advisor AI, for example. All of this can be done with open-source tooling, without expensive proprietary solutions.

In summary, the platform will be built not as a static system but as a **self-evolving organism**. Using open frameworks for RLHF and agent feedback loops, we ensure that both the AI's **core skills** (the model's quality) and its **strategies** (the agent's tactics) improve over time. This creates a virtuous cycle: the longer our AI runs, the smarter and more effective it becomes, continuously compounding its ability to achieve the user's goals (and make money).

Infrastructure and Deployment (Next.js, Netlify, and Free Hosting)

We intend to **host the entire system on free or minimal-cost infrastructure** without sacrificing performance. The key is leveraging serverless and client-side execution cleverly, and using open-source platforms as much as possible:



Conceptual stack for an AI-driven web platform using open-source components. A composable web frontend (e.g. Next.js on Netlify, left) connects to an AI orchestration layer (frameworks like LangChain or LlamaIndex, center), which in turn interfaces with various model and data providers (right). This modular design leverages free open-source tools and services (from LLM providers to vector databases) to build a scalable AI system [24] [23].

- **Frontend (Next.js on Netlify/Vercel):** We'll build a sleek web application using **Next.js**, a React framework, to serve as the user interface for our AI platform. Next.js is open-source and developer-friendly, and crucially it can be hosted on platforms like **Netlify** or **Vercel** on their free tiers. Netlify recently improved support for Next.js features via an open-source adapter [25], meaning we can deploy our site with **zero hosting cost** while still using dynamic routes, server-side rendering, etc. The frontend will allow users (or investors/clients) to interact with the AI – for example, a dashboard to assign objectives to the AI, view progress, and see results. Static content and basic UI interactions are all free to serve and highly scalable on these platforms.

- **Serverless Functions for AI Backend:** Netlify and similar hosts support serverless functions (AWS Lambda under the hood) which we can use to handle AI inference requests or tool calls. For instance, when the AI needs to generate text or run some logic, the frontend can trigger a Netlify Function. There are some **limitations** to consider: free serverless functions have memory and execution time limits (often a few hundred MBs and a max of ~10 seconds runtime per invocation). Running a huge 70B model directly in such an environment may not be feasible. **Workaround:** We can host the actual model on a different free resource or use smaller models for on-demand tasks. One approach is to use **Hugging Face's free cloud** – Hugging Face offers an API and "Spaces" where open models can be deployed (some even with free GPU time in limited quantity). Our Netlify function could call a HuggingFace model inference API for heavy LLM tasks. Alternatively, we leverage the fact that some open models are efficient enough to run on CPU with optimization. For example, OpenAI's GPT-oss-20B is optimized to run on consumer hardware [13], and Mistral's 3B/8B models are made for edge devices [12]. Using 4-bit quantization techniques, we could potentially deploy a 7B-13B parameter model **within the memory constraints** of a serverless function (or on a cheap cloud VM) – albeit with slower response times. We'll choose the path that gives acceptable performance at zero cost. The key is, **the architecture is flexible**: the Next.js frontend doesn't care if the "brain" is on a Lambda, a local server, or a remote API, as long as it can call it. Netlify's own guide notes that AI workloads are often treated as a separate API layer and can integrate with any model provider

(OpenAI, HuggingFace, etc.) through the serverless runtime [26]. In practice, that means we can start free and scale up later if needed by swapping out the backend without changing the frontend.

- **Data and Memory Storage:** For the AI's long-term memory and any databases, we will also stick to free-tier solutions. **Vector databases** (to store embedding vectors for knowledge retrieval) can be hosted via open-source on a free tier of a cloud DB or locally. For example, **Supabase** (an open-source Firebase alternative) has a free tier that, combined with its pgvector extension, can serve as a vector store. There are also fully open-source vector DBs like Chroma or Milvus which we could run on a small server if needed. Netlify doesn't host databases directly, but it allows connections to external DB providers easily [23]. We'll utilize that to connect our functions to whichever free database we choose. Similarly, any file storage (for agent's scratch files or intermediate data) could be done on something like IPFS or a free AWS S3 tier. Essentially, we assemble a **patchwork of free resources** – each component (UI, functions, DB, etc.) on a free plan – to support the overall system. This might require some careful monitoring of usage limits, but it ensures we **pay $0 in infrastructure** while starting out.

- **Scaling Considerations:** Even on free infrastructure, our design will allow us to demonstrate a working product and even handle light user loads. If our project gains traction (and funding), scaling up will be straightforward: Netlify and Vercel can scale to enterprise levels (then paid), and since we use open-source components, we could deploy them on our own servers or cloud instances to handle more intensive workloads. The **important part now** is that we do not let hosting costs hinder development or prototyping. We accept some constraints (like maybe slightly slower response times or limited concurrency on free tiers) initially, in exchange for **learning and improving the system without burn rate**. Once the concept is proven, investors would be more willing to fund dedicated resources – but until then, open-source + free-tier hosting keeps us agile.

Building Our Own Tools from Open Source

One powerful aspect of this approach is that **we aren't beholden to any single vendor** – we can create custom tools by adapting open-source projects. Practically, this means if we need a capability, chances are there is an open project we can start from:

- Need a **custom knowledge graph**? Use an open-source graph database (Neo4j community edition, or an RDF store) and populate it with data using our AI.
- Need **image generation or editing** as part of the system? Deploy **Stable Diffusion** (open-source image model) or its forks on a free GPU instance, and integrate it via API. Many stable diffusion models are freely available for creative visuals.
- Need text-to-speech for an AI assistant voice? **Coqui TTS** and **Mozilla TTS** are open libraries we can run to generate speech audio from text. Likewise, **Whisper** (open-source by OpenAI) can do speech-to-text if voice input is needed. All free.
- For **monitoring and interface**, we can integrate open dashboards. For instance, we might use **Streamlit** (open-source app builder) or **Gradio** to quickly spin up internal dashboards to watch what the AI is doing in real-time, without coding a frontend from scratch.
- If our AI needs to interact with emails, Slack, etc., there are open-source API clients for virtually every service. We can script those interactions rather than pay for a proprietary integration platform.

The philosophy is: **use open-source as LEGO blocks**. We assemble the blocks in novel ways to create a system unique to our needs. And if a block doesn't exist, we build it and then open-source it ourselves, contributing back. This maximizes innovation because we stand on the shoulders of giants – tens of thousands of developers contributing to these projects – instead of reinventing the wheel at great cost.

Crucially, by having our **own custom-built system**, we also own the data and the solution. There's no vendor lock-in, and we can ensure privacy/security by keeping everything under our control. Investors often appreciate this because it means no one can suddenly jack up the price or pull the plug on a critical component of our platform. Every tool can be self-hosted or swapped out if needed. For example, if one vector DB doesn't scale well, we can switch to another open one because we used standard interfaces. If a new, better open-source model comes out in 2026, we can plug it in to replace an older one. **This adaptability is key to long-term growth.**

Real-World Impact and Monetization Strategies

Ultimately, all this advanced tech needs to translate into **real-world revenue**. Simply generating content (blogs, social media posts, etc.) isn't enough for a "system that prints money" – we need to apply the AI to high-value tasks and possibly create new services. Here are some ways our AI platform can directly or indirectly monetize its capabilities:

- **AI Software-as-a-Service (SaaS):** We can offer the platform as a service that businesses pay for. For example, an **AI virtual business consultant**: companies could pay a subscription to have our AI analyze their data, handle customer inquiries, manage their marketing campaigns, etc. Because our system can do research, perform analytics, and even generate or execute code, it could function like an **autonomous employee or team** for a client. Imagine a small business owner delegating their online advertising and website optimization to our AI – the AI might A/B test content, adjust SEO, design marketing materials (via generative models), and manage an online store, aiming to increase sales. This provides tangible economic value that clients would pay for.

- **Automated E-commerce & Content Platforms:** Our AI could run its **own online businesses**. For instance, it might create and manage affiliate marketing websites or dropshipping stores. An early experiment in 2023 showed the promise of this approach: an entrepreneur gave ChatGPT $100 and the directive to make as much money as possible – the AI's plan led to launching an eco-products website ("Green Gadget Guru"), which **generated over $1,000 in a day and was valued at $25k within a week** (according to the creator's tweets) [27] . While that project wasn't sustained long-term, it proved that an AI can set up a web business **incredibly fast**. With our more advanced system, we could multiply this effect: the AI could continuously identify profitable niches, create websites or apps around them, and optimize for revenue (through ads, affiliate links, or selling products). Because all the content and operations are handled by the AI, this could scale to many micro-businesses. Each individual site might only make a small profit, but at scale it "prints money."

- **Trading and Financial Analysis:** With models like DeepSeek's R1 specialized for finance and math [28] , our platform could delve into **algorithmic trading or investment advice**. The AI agents could scrape financial news, analyze market data, and execute trades or recommendations much faster than a human. Open-source financial data libraries (like QuantConnect's LEAN engine) could be plugged in. An autonomous hedge fund AI is ambitious, but even offering AI-driven financial research to fintech firms or investors could be lucrative. The key is that our AI can process huge information (thanks to large context models) and react in real-time, potentially finding arbitrage or trends to capitalize on. (Of course, caution and oversight would be needed in such high-stakes domains, but as a revenue stream it's promising if done right.)

- **AI-Generated Software and IP:** Our platform can build software products (via frameworks like MetaGPT/OpenAgents). We could thus **create and sell software or license AI-generated IP**. For example, the AI might develop a mobile app that fills a market need – we can publish that app and earn revenue from it. Or the AI could design game assets, videos, or other digital products that we sell. Because the AI can work 24/7 and iterate quickly, it might come up with dozens of prototypes, some of which could turn into hits. In essence, the AI itself becomes an *engine of innovation*, generating valuable digital assets at almost zero marginal cost. This flips the script from just content creation to full product creation.

- **Enterprise Solutions and Consulting:** We can also deploy our AI system in enterprise environments for a fee. Many companies are looking for ways to automate their operations with AI but lack the expertise. Our open-source solution could be marketed as a **cost-effective automation agent**. We could customize and fine-tune it for specific industries (e.g. an AI that runs a hotel's pricing and booking adjustments, or one that manages inventory and supply chain for a retailer). Because everything is built on open components, we could even let the client host it on their premises (for security) and charge for integration/support. This could become a B2B SaaS or consulting model where the core product is our AI's capabilities.

It's worth noting that **AI adoption is hitting mainstream**: as of 2025, over *60% of consumers use AI to start their daily tasks* [29]. This means the market is primed for AI-driven services and people are more willing to pay or trust AI solutions. By positioning our platform at the forefront of this wave – an AI that can truly handle complex tasks autonomously – we have a strong pitch for investors and customers alike. We can demonstrate use-cases like a fully AI-run website or a fully AI-managed marketing campaign that *increases ROI by X%*, etc. Those kinds of demos will attract paying users.

Finally, since everything is open source, we have another angle: **community and network effects**. We could allow external developers to build plugins or extensions for our AI platform (similar to how WordPress had an ecosystem). This expands the platform's capabilities without us doing all the work, and it can create a community of contributors. A large community increases the platform's value and could itself be monetized via a marketplace or support contracts. It also impresses investors because it shows widespread adoption potential.

Conclusion

In conclusion, by **exploiting every available open-source technology** – from the most advanced free AI models to flexible agent frameworks and free deployment platforms – we can build an AI-driven system with capabilities that were unimaginable just a few years ago. This platform will provide an **ideal environment for AI (and by extension, our project) to grow and learn**: it is autonomous, self-improving, and backed by a wealth of collective innovation. And importantly, it's designed with monetization in mind at every step, targeting high-value applications beyond simple content generation.

What we've described is not science fiction; all the pieces exist today in open repositories and research labs, waiting to be assembled. The plan is to **integrate these components into a cohesive product** that can attract users and investors through its sheer effectiveness and low cost base. By starting open-source and free, we keep overhead near zero while iterating quickly – a strategy that de-risks the project and appeals to investors (who love seeing traction without heavy burn). Once proven, the sky's the limit: with additional resources, we can scale this system up with more compute and data, further improving the AI's intelligence and capabilities.

In short, this approach gives us the best of both worlds: **cutting-edge AI performance with the freedom to innovate cheaply**. It sets the stage for a venture that can truly earn the title of a "billion-dollar project," built on a foundation that we fully control. By staying adaptive, community-driven, and results-focused, we will give our AI (and this business) the ultimate chance to flourish – to continually get smarter and *"print money"* in a sustainable, real way, powered by open technology. We are essentially creating a powerful engine (the AI) and an efficient vehicle (the system around it) that together can drive into any domain – and disrupt it. It's an exciting time, and with the plan outlined above, we have a roadmap to turn this dream into reality.

Sources: The information and claims made above are supported by recent research and developments in the AI field. Key references include: open-source model capabilities (Meta AI's Llama 4 [1] [4], Alibaba's Qwen-3 [6] [5], DeepSeek [9], Mistral [12] [11], OpenAI's GPT-oss [13]), autonomous agent frameworks and their features (LangChain [14], Auto-GPT's tool use [15], MetaGPT team simulation [16], CAMEL multi-agent collaboration [17], SuperAGI with memory integration [18], Open Agents coding abilities [19]), methods for AI self-improvement (OpenRLHF for training feedback-driven models [20], BabyAGI's learning loop [21]), and deployment considerations (Next.js on Netlify and integration with AI providers [26] [24]). Additionally, real-world examples like the "HustleGPT" experiment illustrate the potential for AI-driven business growth [27]. All these demonstrate that the building blocks for our project are not only available, but have been successfully used in isolation – our task is to combine them into something greater.

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [28] Top 9 Large Language Models as of December 2025 | Shakudo
https://www.shakudo.io/blog/top-9-large-language-models

[14] [15] [16] [17] [18] [19] [21] [29] Top 10 Open-Source AI Agent Frameworks to Know in 2025
https://opendatascience.com/top-10-open-source-ai-agent-frameworks-to-know-in-2025/

[20] [22] GitHub - OpenRLHF/OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework based on Ray (PPO & GRPO & REINFORCE++ & TIS & vLLM & Ray & Dynamic Sampling & Async Agentic RL)
https://github.com/OpenRLHF/OpenRLHF

[23] [24] [26] Building AI experiences on Netlify | Netlify Developers
https://developers.netlify.com/guides/building-ai-experiences-on-netlify/

[25] Next.js on Netlify
https://docs.netlify.com/build/frameworks/framework-setup-guides/nextjs/overview/

[27] How One Guy Tried Using ChatGPT to Launch a Business With Just $100 - Business Insider
https://www.businessinsider.com/how-to-use-chatgpt-to-start-business-make-money-quickly-2023-3