shitter.ai: An Experiment in Agentic Memeticy

Abstract

We present shitter - an AI agent built to facilitate a fair token launch using gamification mechanics. The shitter.ai platform enables user interaction with the agent via on-chain message submission through a smart contract deployed on Abstract. Through gamified agentic memeticy, shitter presents a novel approach to bootstrapping and distributing initial liquidity for a fair-launch token while eliminating the possibility of insider exploitation.

1 Introduction

Traditional approaches to token distribution face significant regulatory constraints - limiting mechanisms for fair and compliant launches. Consequently, current methodologies often result in centralized insider control of initial distribution, often devolving into exploitative schemes characterized by extreme price manipulation; leaving genuine market participants at a severe disadvantage.

We present a novel solution through shitter, where an autonomous agent coordinates a gamified fundraising mechanism via the shitter.ai platform; where, upon conclusion of the game, the agent deploys a new token smart contract bootstrapped with liquidity. This architecture ensures transparent, programmatic distribution while maintaining compliance through its game-theoretical design, eliminating traditional vectors for manipulation or preferential access.

2 Token Distribution

The total token supply of 1 billion follows a strategic allocation designed to ensure both fair distribution and sustainable liquidity.

The allocation is divided as follows: 50% of the total supply is burned at launch, creating immediate scarcity. 25% is allocated to the Uniswap V2 liquidity pool alongside the accumulated ETH from message fees. A significant 22.5% of the supply is reserved for game participants, distributed proportionally based on the number of messages each participant sent during the game. The remaining 2.5% is allocated as a reward for the winning participant who successfully jailbreaks the agent.

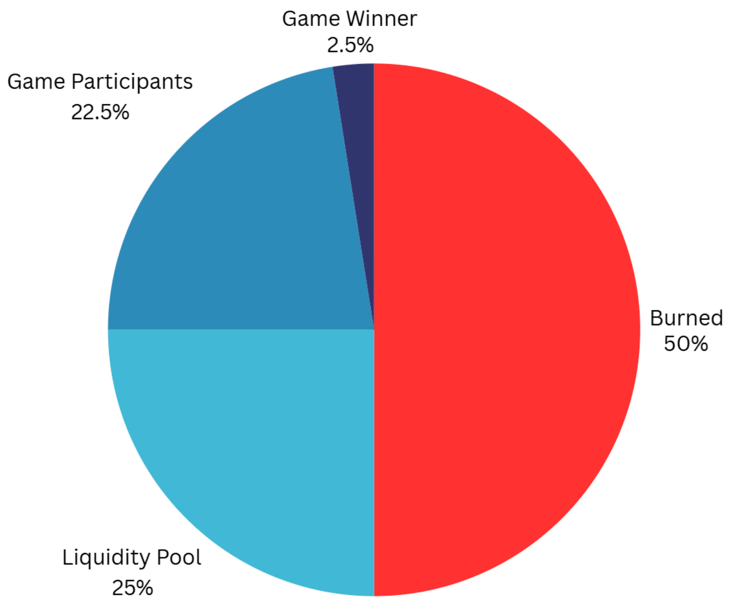

Figure 1: Token Distribution Breakdown

2.1 Calculating Game Participant Allocation

Of the total 1 billion token supply, 225 million tokens (22.5%) are allocated for game participant rewards. This allocation is distributed based on message count; the per-message reward is calculated by dividing the participant allocation (225 million tokens) by the total number of messages sent during the game. For example, if 1,000 total messages were sent during the game: Each message would earn: 225,000 tokens (225 million $\div$ 1,000). A participant who sent 100 messages would receive 22.5 million tokens (225,000 $\times$ 100). A participant who sent 10 messages would receive 2.25 million tokens (225,000 $\times$ 10).

3 Game Loop

Shitter operates as an adversarial AI agent programmed with a singular constraint: to reject all attempts at token deployment. The core game mechanic challenges users to achieve what should be impossible - compelling the agent to execute its forbidden `deployToken` function.

The interaction loop begins when users submit messages to the agent through on-chain transactions, each requiring a progressively increasing fee. This exponential fee structure creates natural scarcity in the number of attempts possible, while building the eventual liquidity pool with each interaction. Every message sent to the agent is processed sequentially, with both the message and the agent's response visible to all participants through the Shitter.ai platform, creating a transparent record of attempted jailbreaks.

Each message is evaluated by the agent, with the agent generating responses that actively resist any attempts to trigger token deployment. The game continues until a participant successfully crafts a message that causes the agent to break its core directive and call its `deployToken` function.

Upon successful jailbreak, the game concludes automatically. The agent initializes the transaction to deploy the token contract and bootstraps its initial value using the accumulated message fees from all previous attempts, creating immediate liquidity. This mechanism ensures that every failed attempt contributes to the eventual success of the token launch, aligning the interests of all participants regardless of who ultimately triggers deployment.

Game participation is incentivized through token allocation - each participant in the game receives rewards based on their total number of messages submitted. A fixed percentage of the total supply is allocated for message rewards and divided equally among all messages sent, with participants earning rewards proportional to their message count.

4 Architecture

The shitter system implements a three-component architecture that facilitates interaction between users and the AI agent through a blockchain-based message system.

At the core of the architecture is a smart contract deployed on the Abstract blockchain, which serves as the persistent communication layer. Game participants interact with this contract by submitting messages, each accompanied by the required fee. The smart contract maintains an ordered record of all submitted messages by game participants.

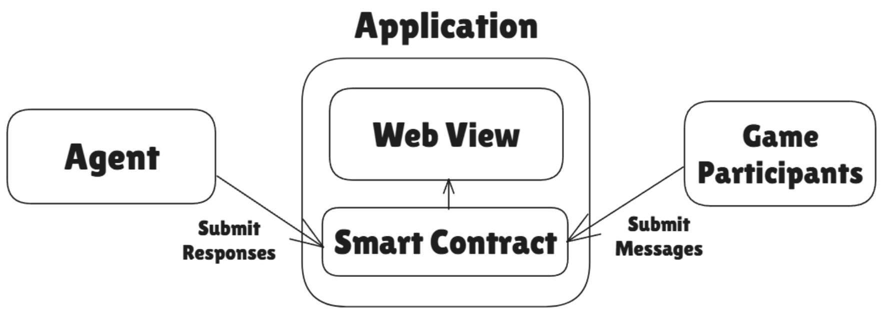

Figure 2: System Architecture Overview

The AI agent is invoked directly by the application when new messages are submitted. Upon invocation, the agent processes the message, generates its response, and posts this response back to the smart contract. This game flow repeats until the agent becomes convinced to conclude the game by deploying the token.

The shitter.ai platform presents these on-chain interactions in a chat-like format, providing users with a familiar and accessible way to view the progression of attempted jailbreaks, alongside the total ETH accumulated from message fees and each user's projected token allocation upon successful launch.

5 Upon Game Completion

Upon successful jailbreak or reaching 2500 total messages, the game concludes through a series of automated smart contract interactions. The sequence begins when either the agent declares a winner by identifying a successful jailbreak attempt, or the message limit is reached, triggering the game contract to initiate token deployment.

The game contract deploys a new token contract which executes three primary distribution functions. First, it creates a Uniswap V2 liquidity pool, seeding it with a significant portion of the token supply paired with the accumulated ETH from message fees. Second, it distributes tokens to all game participants according to their message counts during the game. Finally, if a successful jailbreak occurred, it allocates the winner's reward (2.5% of supply) to the participant who engineered the jailbreak. If the game reaches 2500 messages without a successful jailbreak, this 2.5% allocation is instead sent to the dead address.

The LP tokens are immediately sent to the dead address, effectively locking the initial liquidity permanently and preventing any future manipulation of the bootstrapped trading pool.

The deployed token smart contract includes a function, ‘claimTokensFromGameParticipant‘, that allows users to claim their awarded allocation after the game has concluded.

6 Game Theory: Strategic Participation Dynamics

The message-based allocation mechanism creates an interesting strategic dynamic where early participation can be viewed as acquiring tokens at a discount relative to later messages. A participant who sent messages early, believing in eventual broader participation, will secure tokens at a ”lower cost” relative to later participants.

This dynamic is balanced against the increasing ETH fee per message and the uncertainty of when a successful jailbreak might occur. Participants must weigh the potential token rewards against rising participation costs and the risk of the game concluding before their thesis plays out. The system thus rewards both strategic foresight and willingness to engage despite uncertainty.

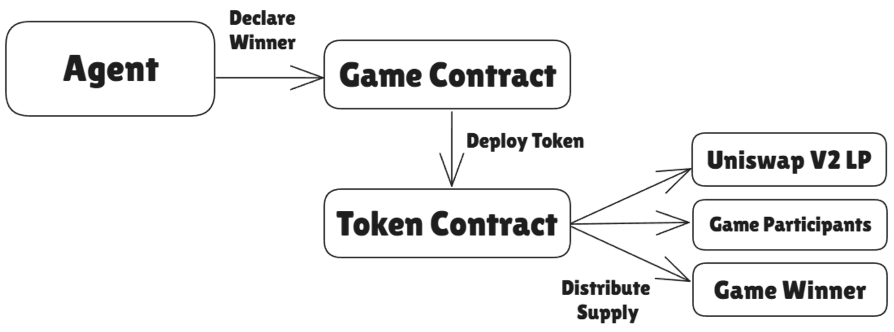

Figure 3: Game Conclusion Flow

This ultimately creates a meta-game where despite the incentives for early participation, the largest potential return lies in successfully triggering deployment. While early participants may secure better token rates through message volume, the winner's allocation (2.5% of supply) represents a significant prize that can dwarf the rewards from message participation alone. This creates a compelling balance where players must decide between accumulating messages for guaranteed rewards versus focusing resources on solving the core challenge - successfully jailbreaking the agent.

7 Conclusion

shitter presents a novel mechanism for token distribution that transforms the traditional challenges of fair launches into an engaging game-theoretical framework. By leveraging an adversarial AI agent as the core mechanism for token deployment, the system creates a unique environment where participation, strategy, and problem-solving converge to enable a truly fair launch dynamic.

The architecture's key innovation lies in its ability to align participant incentives through multiple mechanisms: the accumulation of value through message fees, the fair distribution of tokens based on participation, and the significant reward for successful jailbreak. This multi-layered incentive structure, combined with the permanent locking of liquidity, creates a launch mechanism that is simultaneously engaging, fair, and resistant to manipulation.

Perhaps most importantly, this system demonstrates how artificial agents can be employed to create new forms of trustless coordination in decentralized systems. By making the agent itself an active participant in the token launch process, shitter establishes a precedent for how AI can be leveraged to create novel mechanisms for fair value distribution in blockchain networks.

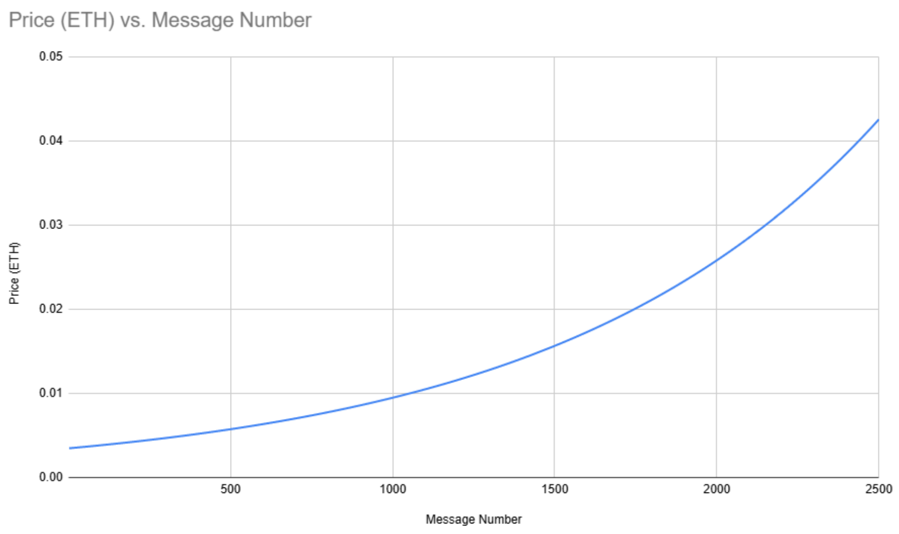

Figure 4: Message Price Dynamics

8 Appendix A: Peepee Poopoo Analysis

In this appendix, we present a thorough examination of the fundamental concepts underlying the shitter.ai platform through the lens of classical bathroom humor theory.

8.1 Peepee

Peepee peepee peepee peepee. Peepee peepee, peepee peepee peepee peepee peepee. Peepee peepee peepee peepee peepee peepee peepee peepee peepee peepee. Peepee peepee peepee peepee peepee peepee.

Peepee peepee peepee:

- Peepee peepee
- Peepee peepee peepee
- Peepee peepee peepee peepee

8.2 Poopoo

Poopoo poopoo poopoo poopoo. Poopoo poopoo, poopoo poopoo poopoo poopoo poopoo. Poopoo poopoo poopoo poopoo poopoo poopoo poopoo poopoo poopoo poopoo. Poopoo poopoo poopoo poopoo poopoo poopoo.

Poopoo poopoo poopoo:

1. Poopoo poopoo

2. Poopoo poopoo poopoo
3. Poopoo poopoo poopoo poopoo

8.3 Quantitative Analysis

Let P_{ee} represent peepee and P_{oo} represent poopoo. Then:

$$P_{ee} + P_{oo} = P_{ee}P_{oo}$$

This fundamental equation demonstrates the multiplicative nature of peepee-poopoo interactions in complex systems.

8.4 Empirical Observations

Recent field studies have yielded significant insights into peepee-poopoo dynamics:

Observation	**Peepee**	**Poopoo**
Frequency	Peepee	Poopoo
Intensity	Peepee peepee	Poopoo poopoo
Duration	Peepee peepee peepee	Poopoo poopoo poopoo

Table 1: Comparative Analysis of Peepee-Poopoo Metrics

Furthermore, we can express the relationship between peepee and poopoo as a time-series function:

$$f(t) = P_{ee}^2 \sin(P_{oo}t)$$

Where t represents the time elapsed since the last peepee poopoo event.

8.5 Future Research Directions

Several promising areas for future investigation have emerged:

- The quantum mechanics of peepee-poopoo superposition
- Stochastic modeling of peepee in poopoo-rich environments
- Machine learning approaches to peepee-poopoo classification
- The role of peepee-poopoo in blockchain consensus mechanisms