Cuttlefish Labs: BuilderAgent & Agent Swarm Architecture Overview

---

1. BuilderAgent Architecture

Purpose: The BuilderAgent is the core autonomous execution layer within the Cuttlefish ecosystem. It interprets market signals and strategic blueprints to perform capital deployment, trading, and liquidity operations on behalf of the DAO.

Key Components: - `TradeSignalMarket`: AI module simulating market intelligence; generates signals such as arbitrage opportunities, LP imbalance alerts, and treasury yield reallocations. - `BuilderAgent.run()`: Receives blueprint and signals; determines actions (e.g. swap, stake, rebalance); logs every action into TrustGraph. - `vault_balance`: Simulated treasury logic, tracking available capital for on-chain execution.

Example Decisions: - Provide/withdraw LP based on impermanent loss signals - Execute trades on Uniswap via CWALayer - Adjust stablecoin allocations for risk mitigation

---

2. TrustGraph / Vault / FusionSwarmFactory

TrustGraph: - Tracks agent decisions and their justifications as verifiable data nodes - Enables lineage, accountability, and ethical review - Stores causal links between blueprints, signals, and actions

CuttlefishVault (Contract to be finalized): - Treasury custody layer, allowing BuilderAgent access with permissions - Executes on-chain via `vault.executeTradeOnUniswap()` or equivalent - Multi-signature and proposal-gated for withdrawals

FusionSwarmFactory: - On-chain orchestration hub - Emits AgentDeployed, AgentActionExecuted, AgentPredicted - Provides `addLiquidity`, `swapTokens`, `submitFuturePrediction` - Integrates with Chainlink, Uniswap, and MCP interfaces

---

3. Agent Memory & Toolchain Evolution

Evolution Path: - v1: Static simulation agents (pre-coded signal responses) - v2: Blueprint + Signal driven agents with persistent vault logic - v3: TrustGraph-augmented agents with memory, ethical gating, and AI self-reflection

Pipeline: - Python AI core → CWALayer (TS) → Solidity executor - Chainlink Functions for prediction checking - zkML future hooks for model attestations

---

4. Solidity + Web3 Role for Salmaan

Primary Responsibilities: - Finalize and secure `CuttlefishVault.sol` - Enhance `FusionSwarmFactory.sol` with real DEX integration (Uniswap v4, Balancer) - Connect CWALayer to BuilderAgent Python backend for real-time agent calls - Deploy contracts to testnet with Ethers.js/Hardhat suite - Implement `submitFuturePrediction()` validation for FutureBench

Roadmap: - Week 1: Testnet deploy Vault + Factory + basic Agent - Week 2: Add liquidity logic + prediction eval - Week 3+: Iterate with Chainlink / zk attestations / DAO triggers

---

Prepared for Salmaan by Cuttlefish GPT