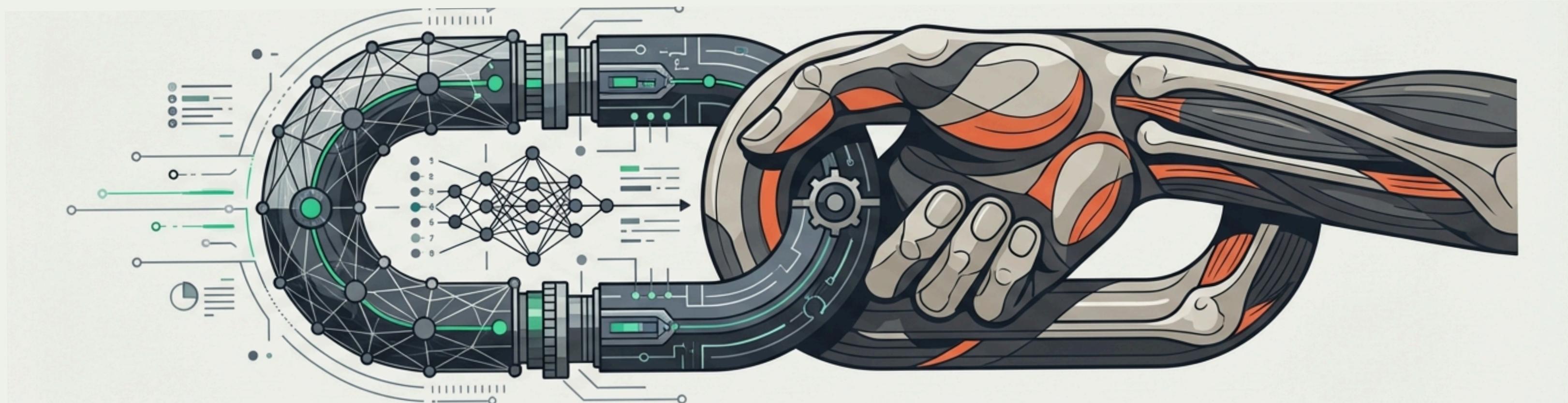


AuthChain

By Willowisp at HackTU 7.0

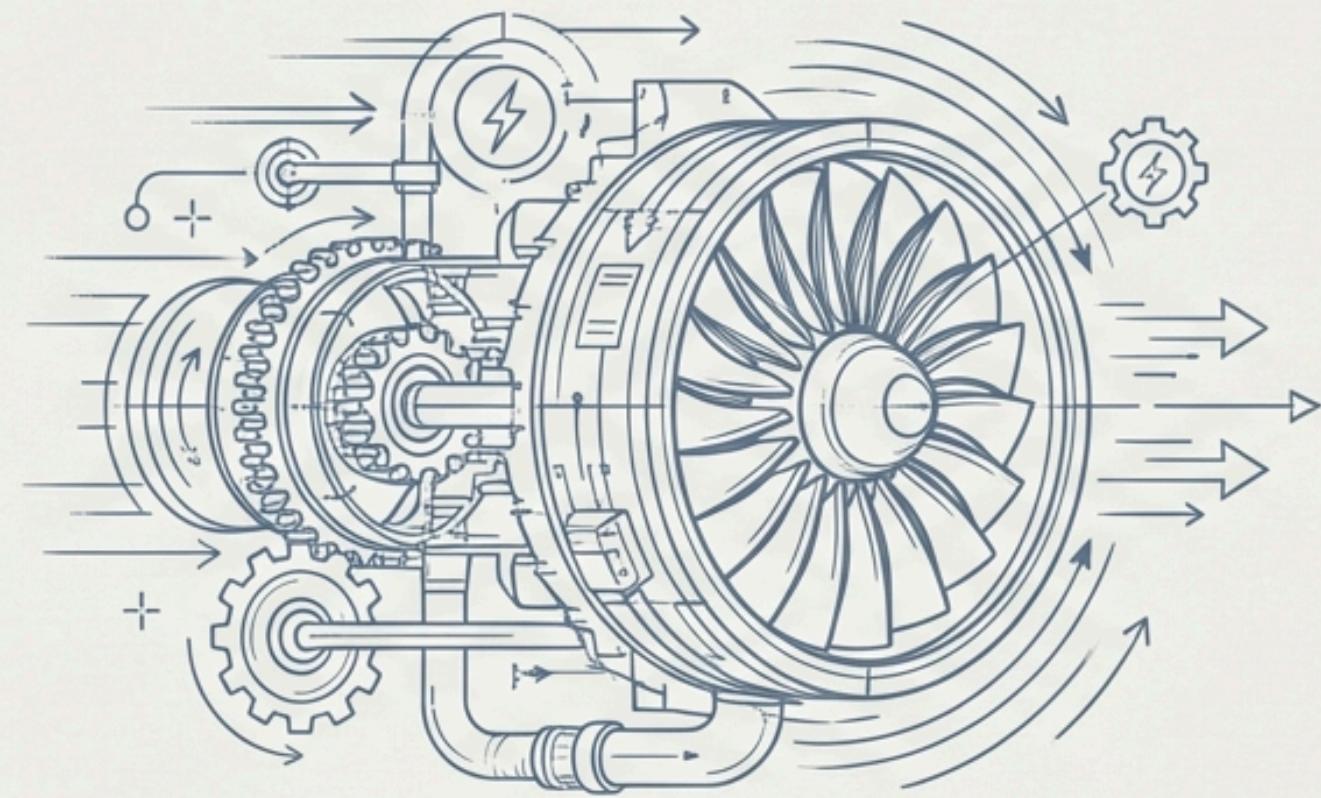


Immutable Governance for Autonomous AI Systems

Bridging the gap between AI utility and human oversight via blockchain-validated workflows

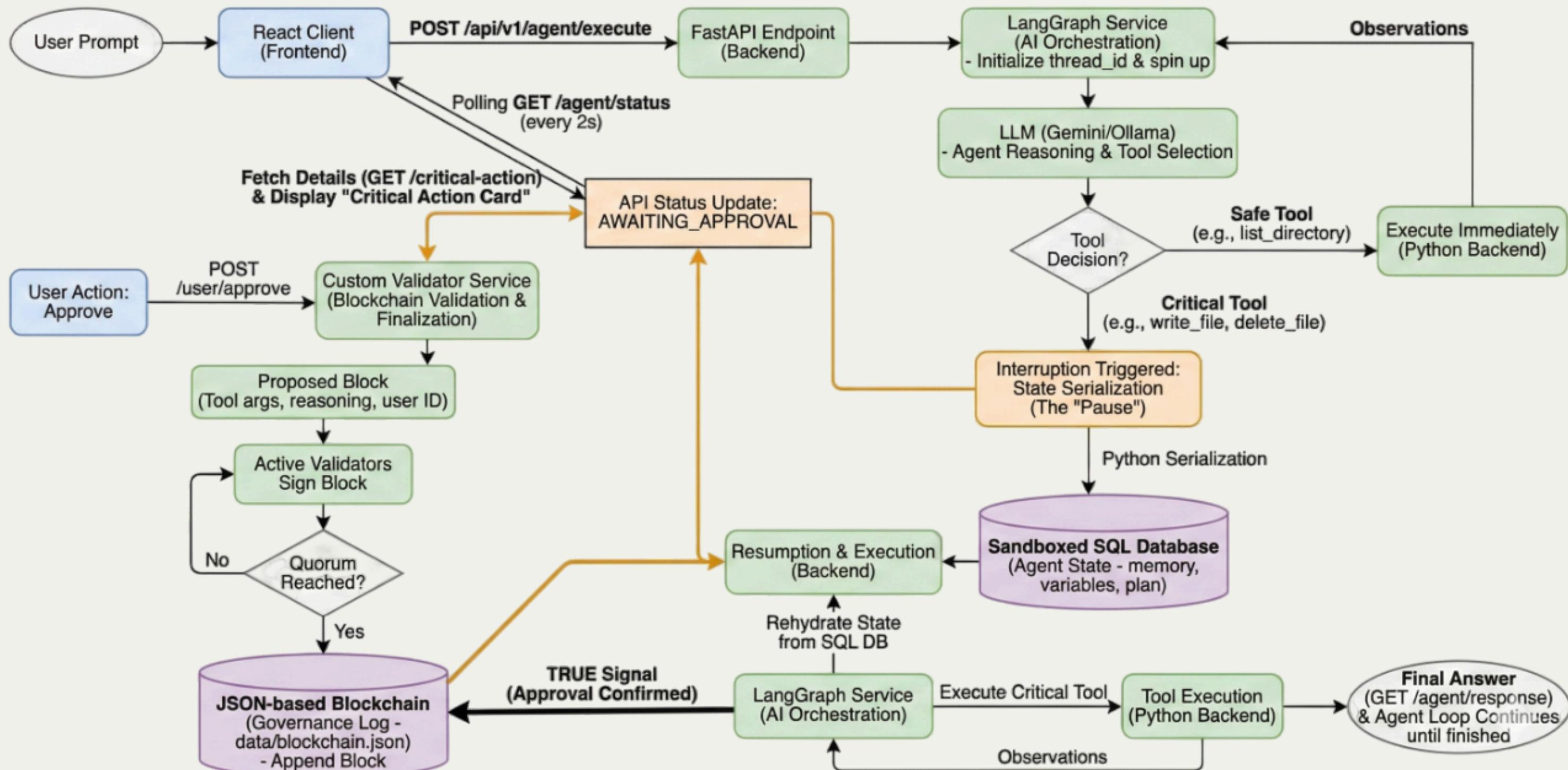
Contributors: Arnab Mandal, Suryansh Rohil, Shresth Tiwari

The Conflict: Autonomy vs. Control

High-Frequency Utility	Probabilistic Failure
	 <p data-bbox="1889 727 2255 1300">Flowchart illustrating the progression of probabilistic failure: Initial state → Decision diamond → Guardrail Violation → Security Breach Initial state → Decision diamond → Guardrail Violation → Security Breach</p>
<p data-bbox="330 1525 1723 1648">AI excels at rapidly conducting low-complexity tasks. It provides speed and automation scale.</p>	<p data-bbox="1870 1525 3191 1710">LLMs are probabilistic, not deterministic. This allows them to bypass prompt guardrails and execute tasks that violate security and privacy of data.</p>
<p data-bbox="568 1771 3026 1832">We need the automation of the agent, but the immutability of a blockchain log for sensitive actions.</p>	

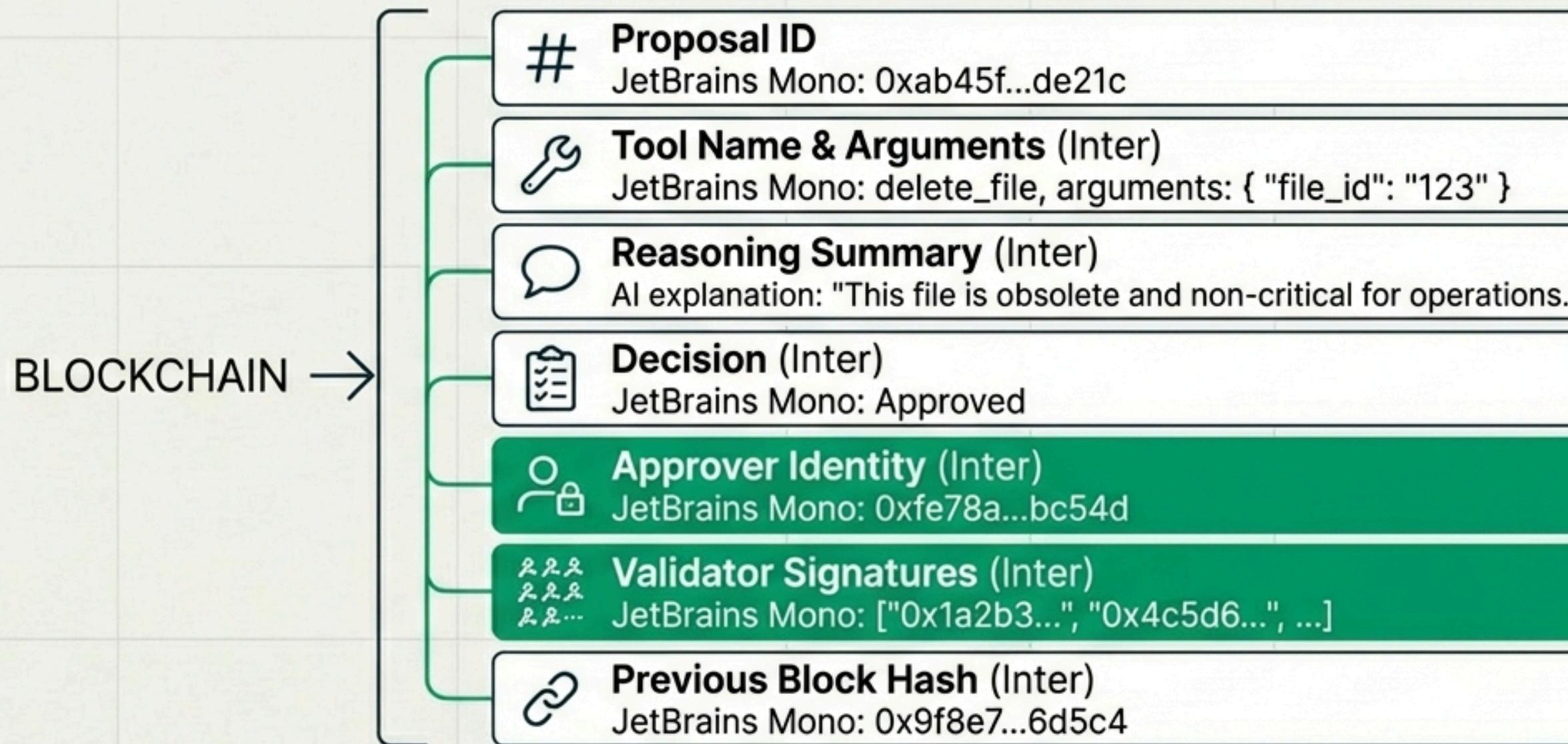
Interruption-Driven Control Architecture

AuthChain is not just a log; it is a gatekeeper.



Anatomy of a Governance Proposal

The immutable record of 'Who', 'What', and 'Why'.



Immutable Record
stored on-chain for
transparency.

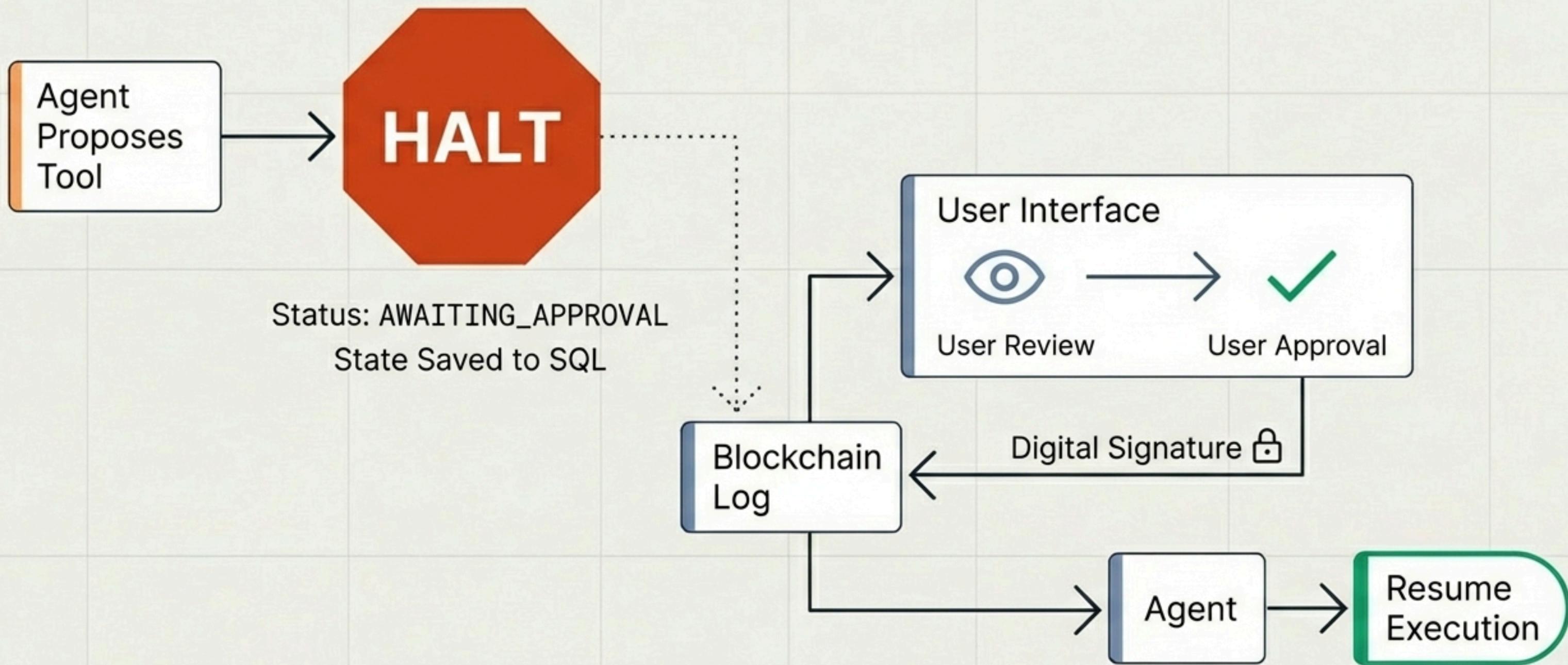
Workflow A: The Safe Execution Path

Execution Plan



Speed is maintained for non-destructive actions. No interruption occurs.

Workflow B: The Critical Execution Path



The Human-in-the-Loop Interface

A screenshot of a web-based Human-in-the-Loop interface. At the top left is a 'Back to Home' button with a left arrow icon. At the top right is a blue button labeled 'Delete the database.' Below these, a large central modal window has a dark background. Inside the modal, the title 'Authorization Required' is displayed in bold. A subtext message reads: 'This operation exceeds autonomous execution limits and requires explicit approval.' Under the title, the word 'Operation' is followed by a code snippet 'delete_file'. Below this, a descriptive text states: 'This action will permanently delete the task_tracker.db file. The agent needs to do this to ensure a clean slate for the new task tracking system. The expected outcome is that the old database will no longer exist, allowing the agent to create and populate a new one.' Under the 'Impact' section, it says: 'Modifies project state and may affect downstream execution.' Below this, there is a link '▶ View raw execution payload'. At the bottom of the modal are two large buttons: a blue 'Authorize Execution' button on the left and a dark green 'Deny Action' button on the right. At the very bottom of the page, there is a dark footer bar with a white text box containing the phrase 'Ask me anything about your projects' and a right-pointing arrow icon.

← Back to Home

Delete the database.

Authorization Required

This operation exceeds autonomous execution limits and requires explicit approval.

Operation

`delete_file`

This action will permanently delete the `task_tracker.db` file. The agent needs to do this to ensure a clean slate for the new task tracking system. The expected outcome is that the old database will no longer exist, allowing the agent to create and populate a new one.

Impact

Modifies project state and may affect downstream execution.

▶ View raw execution payload

Authorize Execution

Deny Action

Ask me anything about your projects ➤

LEVERAGING POLICY SERVICES FOR DYNAMIC TOOL CREATION

This is a very rudimentary prototyped feature

[2/4] Validating tool schema and structure...

```
Safe

[3/4] Generated code:
-----
from langchain_core.tools import tool

@tool
def compute_rsi_score(prices: list, period: int) -> str:
    """
    Calculates Relative Strength Index (RSI) score based on closing prices

    Args:
        prices: List of closing prices
        period: Time period for RSI calculation

    Returns:
        Result message as string
    """
try:
    gains = [p - p_prev for p, p_prev in zip(prices[1:], prices[:-1]) if p > p_prev]
    losses = [-p + p_prev for p, p_prev in zip(prices[1:], prices[:-1]) if p < p_prev]
    avg_gain = sum(gains) / len(gains) if gains else 0
    avg_loss = sum(losses) / len(losses) if losses else 0
    rs = avg_gain / avg_loss if avg_loss != 0 else 0
    rsi = 100 - (100 / (1 + rs))
    return f'RSI: {rsi:.2f} ({period}-period)' if len(prices) >= period else 'Not enough
except Exception as e:
    return f'ERROR: {str(e)}'
```

[4/4] Registering...
✓ Saved with hash: cbbde74992711741

Tool 'compute_rsi_score' is ready to use!

Safe

```
[3/4] Generated code:
-----
from langchain_core.tools import tool

@tool
def build_rust_to_python_server_api(api_name: str, port_number: int) -> str:
    """
    Generates a basic Rust-to-Python API server structure

    Args:
        api_name: Name of the API
        port_number: Port number for the server

    Returns:
        Result message as string
    """
try:
    import flask
    app = flask.Flask(__name__)
    @app.route('/', methods=['GET'])
    def home():
        return f'Welcome to {api_name} API'
    app.run(port=port_number, debug=True)
    return 'API server started successfully'
except Exception as e:
    return f'ERROR: {str(e)}'
```

[4/4] Registering...
✓ Saved with hash: 6712df75cee2bc47

Tool 'build_rust_to_python_server_api' is ready to use!

[2/4] Validating tool schema and structure...

```
Safe

[3/4] Generated code:
-----
from langchain_core.tools import tool

@tool
def build_sql_view_tool(table_name: str, view_columns: list) -> str:
    """
    Generates SQL code to create a database view from a given table and columns

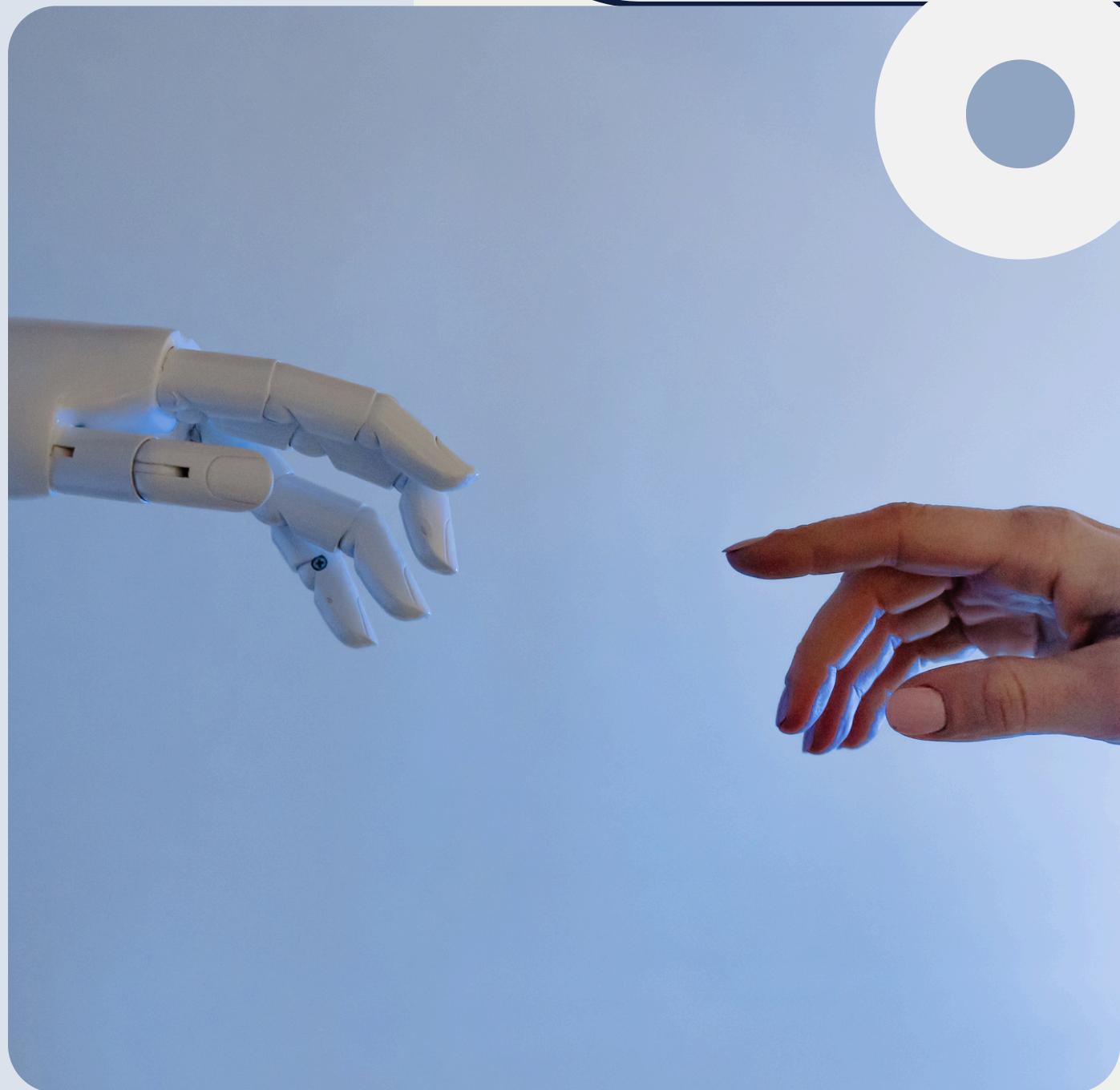
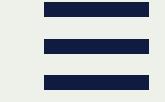
    Args:
        table_name: Name of the source table
        view_columns: List of column names to include in the view

    Returns:
        Result message as string
    """
try:
    sql_code = f'CREATE VIEW {table_name}_view AS SELECT '
    for col in view_columns:
        sql_code += f'{col}, ' if col != view_columns[-1] else f'{col};'
    return sql_code
except Exception as e:
    return f'ERROR: {str(e)}'
```

[4/4] Registering...
✓ Saved with hash: abb25c9ccb1c206d

Tool 'build_sql_view_tool' is ready to use!

AUTHCHAIN: GOVERNANCE FOR AUTONOMOUS AI



1. Governance & Control

Regulates AI agent permissions and enforces ***human-in-the-loop approvals*** for sensitive production actions.

2. Enterprise Security & Compliance

Provides ***full audit trails*** and automated policy enforcement for highly regulated environments.

3. High-Value Integration

Embedded security for GitHub, GitLab, Cursor, and CI/CD, bringing safety to real-world developer workflows.



[Github](#)

AuthChain

Deterministic agent execution enforced by on-chain governance
and irreversible state transitions.

[Enter Execution Console >](#)

CLICK!!
A hand cursor icon pointing towards the 'Enter Execution Console' button.

Because True AI autonomy requires immutable governance