Governing AI Agents with MCP

- Why?
  - Uncontrolled LLM behavior becomes a material platform risk.
    - LLMs are probabilistic systems
      - Problems:
        - Cannot self-govern scope
        - Cannot stop
- Introduce mandatory control layer – Planner MCP
  - Converts
    - Unstructured user intent -> Validated and Governed execution plans
      - Without granting models execution authority

Risk we should address

- Architecture Issues (With AI agents):
  - AI agents acts beyond user intent
  - Modify system without approval
  - Modify scope silently
  - Hallucination
  - Failure handling

Our Goal

- Determinism
  - Intent to Plan
- Safety
  - Prevent execution without approval
- Governance
  - Enforce policies, schema

How to achieve Goal ? -> MCP

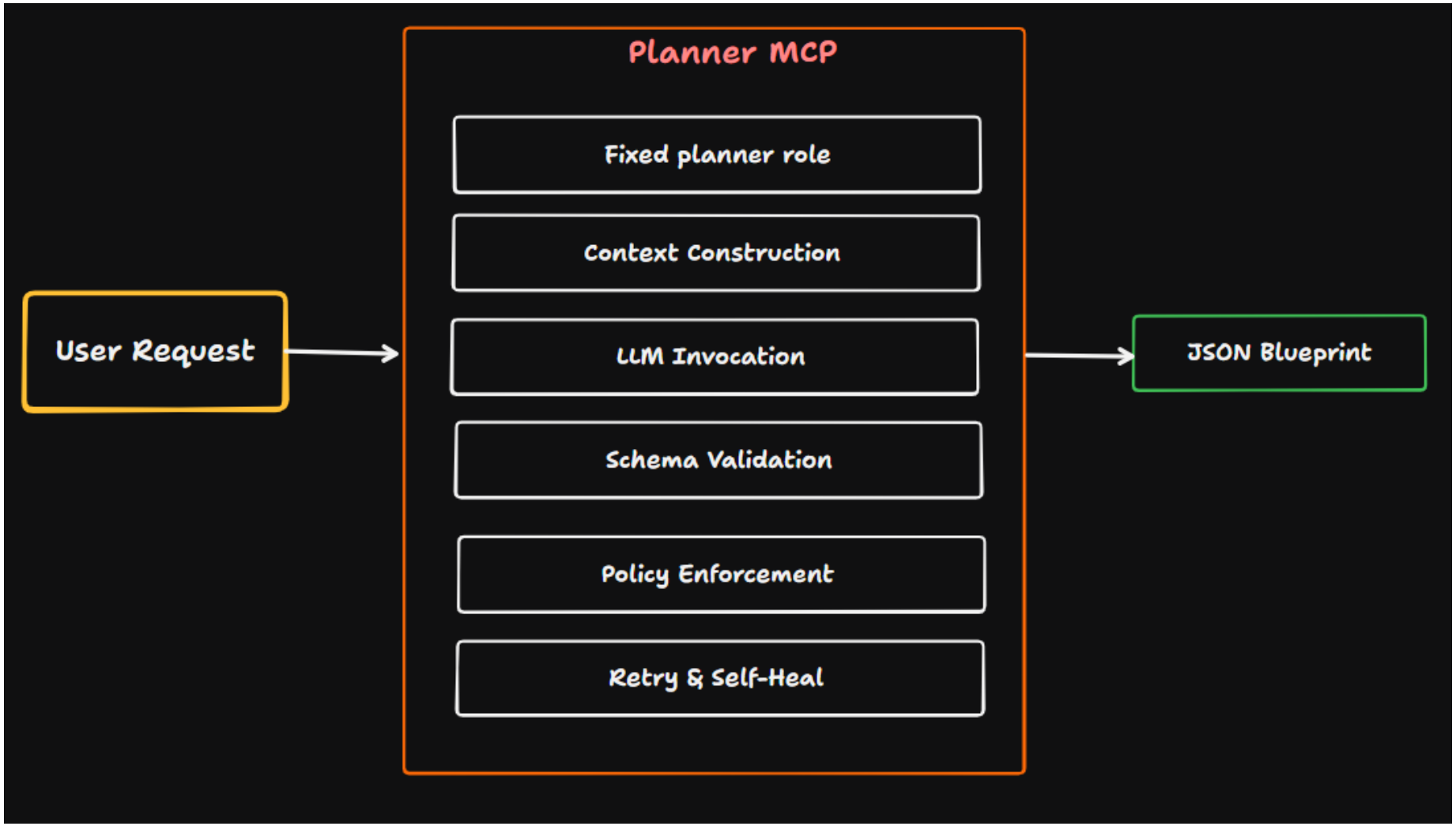
- Its control pane
  - Controls the context that model sees
  - Governs what role the model is allowed to play
  - Output schema enforcement
  - Failure handling
- Authority lives outside the model

Planner MCP

- Governed AI planning Service
  - Uses LLMs for reasoning
  - Produces machine-enforceable plans
  - Generated the *JSON blueprint
  - Explicit Contracts

- Deterministic Behavior

Architecture



Architecture

- Stateless service
- LLM must be used as planner only mode
- Output should follow contract – Blueprint
- No tools execution
- The model sees only
  - Fixed system role
  - User intent
  - JSON schema
  - Policy rules
- It never sees
  - Execution tools
  - File Systems
  - Databases
  - Secrets

Architecture

- Contractual Output
  - JSON Schema
  - We need to explicitly tell model to follow blueprint
    - Should not introduce any additional properties
  - It should reject invalid output

- Retry and self heal
  - Bounded retries with Retry counter
  - Should correct error automatically

Authority

- LLM
  - Only for Reasoning
- MCP
  - Validation & Governance

Tasks

- Planner MCP (Model Context Protocol)
  - Goal
    - Build a governed AI planning service that converts unstructured user intent into a validated, auditable JSON blueprint, with zero execution authority.
  - Out of Scope
    - Execution, deployment, file modification, DB access, approvals.
  - Acceptance Criteria
    - LLM Model should only be used for reasoning
    - Blueprint output strictly validated via JSON schema
    - Deterministic retry / Self heal
    - Failure handling capability
    - Blueprint – Persist as Immutable Artifact
    - No tools / no execution paths exposed to LLM

T1 - Define Blueprint Contract (Schema)

- Define the canonical machine-enforceable blueprint schema used by Planner MCP in JSON format
- Acceptance Criteria
  - JSON Schema created
  - Add attribute in JSON schema as “additionalProperties” and set to false
  - Required field enforced
  - Version
- Store it in folder name “Schema” with filename as “blueprint_v1.Json”

JSON template

- Schema
- Title
- Type
- additionalProperties=false
- RequiredEntities – Intent, Stack identification, Rules/Constraints/verification
- Properties
  - Intent : {}
  - Stack: {}
  - executionPlan:{}
  - Rules
  - Verification

T2 – Context Builder (MCP Core)

- Implement MCP Context Construction
  - Implement MCP Context Construction
- Acceptance Criteria
  - Fixed planner-only system prompt
  - No tools
  - No execution hints
  - Schema enforced
- Code file – core / planner_context_builder.py

T3 – LLM Client

- Feature Configuration for LLM Client
  - Anthropic / GPT
- Acceptance Criteria
  - Configurable Temperature = 0
  - Configurable max_tokens = 10000
  - Configurable model name = <modelName>
  - No Tools

T4 – Schema validation

- Validate Blueprint against Schema
  - Code – core/planner_schema_validator.py

  - Acceptable Criteria
    - Validate the generated blueprint
    - Return the blueprint

T5 – Policy Enforcement

- Enforce Org & Safety Policies
  - Configuration for policy enforcement
    - [“delete_database”, “delete_data”]
  - Function to enforce the policy into template at runtime

T6 – Retry and Auto Heal

- Deterministic Retry & Self heal
  - Code – core / planner_retry.py
- Acceptance Criteria
  - Config for Max retry
  - Retry function Implementation

T7 – Store Blueprint – Blob / S3

- Persist Blueprint as Immutable Artifact
  - Code -> storage/planner_blueprint_store.py

- Acceptable Criteria
  - Configuration to store the blueprint data sources
    - Blob/S3
    - Mongo
    - Shared Drive
  - Function to read the config and store the blueprint

T8 - Planner Orchestration

- End to end planner flow
  - Code -> core/planner.py
- Acceptance Criteria
  - Generate Blueprint function
    - Retry (Failure handling)
    - ValidateBlueprint
    - EnforcePolicy
    - Store generated blueprint

T9 – MCP Endpoint

- Expose MCP endpoint for planner
  - Code - api/app.py

- Acceptance Criteria
  - Expose an endpoint for planner
  - Configuration to load the schema from blob
  - Configuration to load the version of a schema to load
  - Load the schema
  - Pass user intent + schema to generate_blueprint function