# **Technical Design Specification: Project Manager Agent Implementation**

Version: 10.0

Date: 18 June 2025

Status: Proposed

## **1. Architectural Overview**

This document specifies the technical implementation for the initial version of the Project Manager (PM) agent. This implementation focuses on the PM's role as an Auditor, as defined in TRR v27.0. A new agent graph, build\_pm\_agent\_graph, will be created to handle the auditing logic. The supervisor UI will be updated with a new modular test button to trigger the PM agent and verify its functionality using the "Hello, World!" test artifacts.

## **2. State Management Modifications (**agent\_core\_v9.py**)**

To support the PM agent, the central GraphState will be expanded with three new keys:

* pm\_goal: Optional[Dict]: Will hold the goal artifact (e.g., a design\_synthesis.json) that the PM is auditing against.
* pm\_evidence: Optional[Dict]: Will hold the evidence artifact (e.g., a coding\_execution\_report.json) that the PM is auditing.
* pm\_audit\_report: Optional[Dict]: Will store the final structured JSON output from the PM agent.

## **3. Computational Graph: PM Agent (**agent\_core\_v10.py**)**

A new graph will be created to encapsulate the PM's logic.

* Function: build\_pm\_agent\_graph()
* Structure: This will be a simple graph with a single node, generate\_pm\_audit\_node, to ensure robustness.
* Node Logic (generate\_pm\_audit\_node):
  1. The node will retrieve pm\_goal and pm\_evidence from the GraphState.
  2. It will use a new prompt, create\_pm\_audit\_prompt, formatting it with the stringified JSON content of the goal and evidence artifacts.
  3. It will invoke the LLM to perform the audit.
  4. It will parse the JSON response from the LLM and store it in the pm\_audit\_report key in the GraphState.

## **4. New Prompt Specification (**prompts\_v9.yaml**)**

A new, detailed prompt is required to guide the PM agent's auditing task.

* Prompt Name: create\_pm\_audit\_prompt
* Content:

create\_pm\_audit\_prompt: >

You are an automated Quality Assurance and Project Management agent. Your task is to audit an evidence artifact against a goal artifact to verify task completion.

\*\*Instructions:\*\*

1. Analyze the `GOAL` artifact, which specifies the intended work.

2. Analyze the `EVIDENCE` artifact, which represents the work that was done.

3. Compare the `new\_files` list in the `GOAL` with the `final\_code\_artifacts` list in the `EVIDENCE`.

4. For every file in the `GOAL`, you MUST verify that a corresponding file exists in the `EVIDENCE`.

5. For every file, you MUST verify that the `content` is identical.

6. Based on your audit, determine if the status is `SUCCESS` or `FAILURE`. The status can only be `SUCCESS` if all files in the goal are present in the evidence with the exact same content.

7. Provide a brief, one-sentence `reasoning` for your verdict.

8. Your entire output must be a single, valid JSON object conforming to the schema below.

\*\*JSON Schema:\*\*

{{

"audit\_summary": {

"goal\_spec\_version": "<The design\_spec\_version from the GOAL metadata>",

"evidence\_report\_version": "<The report\_version from the EVIDENCE metadata>"

},

"status": "<'SUCCESS' or 'FAILURE'>",

"reasoning": "<A one-sentence explanation for the status.>"

}}

### GOAL ARTIFACT:

```json

{goal\_json}

```

### EVIDENCE ARTIFACT:

```json

{evidence\_json}

```

## **5. Supervisor UI Modifications (**supervisor\_v10.py**)**

The test harness in the sidebar will be updated with a new button for the PM agent.

* New Button: st.button("Test PM Agent ('Hello World' Audit)")
* Logic:
  1. On click, the application will load test\_data/test\_design\_synthesis.json.
  2. It will also load test\_data/test\_coding\_execution\_report.json (a new test file to be created).
  3. It will initialize a GraphState, populating pm\_goal with the design synthesis and pm\_evidence with the coding report.
  4. It will set the run\_phase to "PM\_AUDIT".
  5. The application will st.rerun(), triggering a new state machine branch that invokes the build\_pm\_agent\_graph.

## **6. New Test Artifact**

A new file is required to serve as the "evidence" for the PM's unit test.

* File: test\_data/test\_coding\_execution\_report.json
* Content: This file will be a clean, perfect representation of the output from the "Hello, World!" coding test.