# **Technical Design Specification: A Modular Test Harness**

Version: 9.0

Date: 18 June 2025

Status: Proposed

## **1. Architectural Overview**

This document specifies the technical implementation of the Modular Testing Framework proposed in TRR v26.0. The core of this implementation is a "test harness" built into the supervisor\_v8.py UI. This harness will allow a developer to bypass the full pipeline and inject a hand-crafted test file directly into a specific agent (either the Design Agent or the Coding Agent). This will be accomplished by adding a new section to the Streamlit sidebar for triggering these modular test runs.

## **2. File and Directory Structure**

A new directory will be created in the project's root to store the testing artifacts.

* test\_data/: This new directory will contain all hand-crafted JSON files used for modular testing.

## **3. Test Artifact Specifications**

Two initial "Hello, World!" test files will be created and placed in the test\_data/ directory.

### **3.1. Test Input for Coding Agent**

* File: test\_data/test\_design\_synthesis.json
* Purpose: To serve as a perfect, hand-crafted "work order" for testing the Coding Agent in isolation.
* Schema:

{

"metadata": {

"design\_spec\_version": "0.1-test",

"generation\_date": "YYYY-MM-DD"

},

"high\_level\_goal": "Create a simple 'Hello, World!' Python script.",

"file\_modifications": [],

"new\_files": [

{

"file\_path": "hello.py",

"description": "A simple Python script that prints 'Hello, World!' to the console.",

"content": "print(\"Hello, World!\")"

}

]

}

### **3.2. Test Input for Design Agent**

* File: test\_data/test\_research\_synthesis.json
* Purpose: To serve as a perfect, conceptual input for testing the Design Agent in isolation.
* Schema:

{

"metadata": {

"report\_version": "0.1-test",

"generation\_date": "YYYY-MM-DD"

},

"high\_level\_summary": "The goal is to produce a simple Python script that prints 'Hello, World!'",

"key\_architectural\_principles": ["Simplicity", "Standard library only"],

"system\_components": [{"name": "hello.py", "description": "A single Python file to be created."}],

"workflow\_stages": ["Code Generation"],

"state\_management": {"definition": "N/A", "keys": []},

"required\_tools": []

}

## **4. Supervisor UI Modifications (**supervisor\_v8.py**)**

The primary change will be the addition of a new "Modular Testing" section in the Streamlit sidebar.

### **4.1. UI Implementation**

* A new header, st.header("🧪 Modular Testing"), will be added to the sidebar.
* Two new buttons will be created below this header:
  1. st.button("Test Design Agent ('Hello, World')")
  2. st.button("Test Coding Agent ('Hello, World')")

### **4.2. Logic Implementation**

* On "Test Design Agent" click:
  1. The application will read and parse test\_data/test\_research\_synthesis.json.
  2. It will initialize the GraphState with only the a2a\_output key populated with the content of the test file.
  3. It will set the run\_phase directly to "DESIGN\_GENERATION".
  4. The application will then st.rerun(), starting the execution at the Design Agent and completely bypassing the Research phase.
* On "Test Coding Agent" click:
  1. The application will read and parse test\_data/test\_design\_synthesis.json.
  2. It will initialize the GraphState with only the design\_synthesis\_json key populated with the content of the test file.
  3. It will set the run\_phase directly to "CODING".
  4. The application will st.rerun(), starting the execution at the Coding Agent, bypassing both the Research and Design phases.

## **5. Tool Modifications (**supervisor\_v8.py**)**

To ensure the read\_file tool can locate these new test files, its search logic will be updated.

* read\_file Tool: The tool will be modified to search for a requested file in a specific order of priority:
  1. The root project directory.
  2. The test\_data/ directory.
  3. The agent\_outputs\_v8/ directory (for generated artifacts).

This design provides a clean, user-friendly way to conduct isolated, modular tests on the agents, which will significantly accelerate debugging, improve reliability, and enable the development of the Project Manager agent on a stable foundation.