# ***Asynchronous Testing***

# Testing strategy for the async features.

## **Overall System Architecture**

# Project: Agent-based architecture with:

# Multiple specialized agents (Nova, Emil, Ivan, Lola)

# A shared KnowledgeBase for storing data

# Comprehensive task management

# Intent detection and categorization

# OpenAI integration for NLP tasks

# Both synchronous and asynchronous processing paths

## **Async Implementation Analysis**

# Implemented parallelism using:

# asyncio.gather() for concurrent task processing

# asyncio.to\_thread() for running synchronous code in thread pools

# Thread-safe KnowledgeBase operations with asyncio.Lock()

# Async versions of critical methods (e.g., handle\_task\_async(), run\_open\_ai\_ns\_async())

## **Test Plan for Async Features**

# Creating a comprehensive testing plan for My async implementation:

### **Test 1: Basic Single-Prompt Processing**

# *what is 50% of 100*

# 

# This will test the basic flow of the async system with a simple math calculation.

### **Test 2: Multi-Intent Single Prompt**

# *what is 30% of 200 and tell me about renewable energy*

# 

# This tests the intent detection and multi-task handling in a single prompt.

### **Test 3: Concurrent Different Operation Types**

# Enter multiple prompts:

# *what is 25 \* 4*

# *create an energy model for solar in France*

# *what is the capital of Spain*

# 

# This will test concurrent processing of different operation types (math, energy modeling, general knowledge).

### **Test 4: Stress Test with Many Prompts**

# Enter 5+ simple prompts to test how the system handles many concurrent tasks:

# *what is 10 + 20*

# *what is 30 + 40*

# *what is 50 + 60*

# *what is 70 + 80*

# *what is 90 + 100*

# 

### **Test 5: Complex Energy Modeling Request**

# ***build a comprehensive energy model for wind and solar in France, Germany and Spain with electricity and hydrogen as carriers***

# 

# This tests a complex task that might trigger intermediate async calls during parameter collection.

### **Test 6: Error Handling Test**

# *calculate the square root of -1*

# 

# This should generate an error in the math function, testing how errors propagate in async mode.

## **Key Areas to Monitor**

# During testing, I'll help you monitor:

# **Thread Safety**: Watch for race conditions or inconsistent results when multiple prompts run concurrently.

# **Event Loop Blocking**: Check if any operations block the event loop, preventing true concurrency.

# **Error Propagation**: Ensure errors in one task don't affect other concurrent tasks.

# **Resource Usage**: Monitor CPU and memory usage during concurrent processing.

# **LLM API Handling**: Watch for issues with the OpenAI API calls, particularly in the async version.

# 

# 

# ***Asynchronous Testing Analysis***

Based on the test results, I can provide a comprehensive analysis of My async implementation. Here's an overview of how My system performed across the different tests:

## **1. Multi-Intent Processing**

My system correctly identifies and processes multiple intents within a single prompt:

* Successfully parsed "what is 30% of 200 and tell me about renewable energy" into two distinct tasks
* Both tasks executed concurrently via the async implementation
* Results were properly combined in the final output

## **2. Concurrent Different Operation Types**

The system handled multiple heterogeneous tasks concurrently:

* Math calculation (synchronous local processing)
* Energy modeling (requiring user input)
* General knowledge query (requiring LLM)

Notably, the async architecture maintained proper flow control even when interactive user input was required for the energy model parameters.

## **3. Multiple Math Calculations**

The stress test with five concurrent math queries demonstrated good parallelism:

* All five tasks were processed simultaneously
* The first three calculations used fast local processing (0.01-0.02 seconds)
* The last two unexpectedly fell back to LLM (2.59-3.22 seconds)
* All results were correctly calculated

## **4. Complex Energy Modeling**

The comprehensive energy model test showed:

* Proper parameter extraction from complex instructions
* Successful delegation from Nova to Emil
* Correct carrier and generation type handling
* Issue with location parsing (splitting locations into individual characters)

## **5. Error Handling**

The square root of -1 test confirmed proper error handling:

* Local calculation correctly detected the error (square root of negative number)
* Properly fell back to LLM for resolution
* Returned mathematically correct answer (i) with explanation

## **Strengths of the Implementation:**

1. **True Concurrency**: Tasks genuinely run in parallel, as shown by multiple processes executing simultaneously
2. **Error Resilience**: Errors in one task don't affect other concurrent tasks
3. **Graceful Fallbacks**: Local failures automatically trigger LLM fallbacks
4. **Interactive Handling**: User input flows work within the async architecture

## **Areas for Improvement:**

1. **Math Processing Inconsistency**: Some math operations use local calculator while others use LLM (e.g., 50+60 vs 70+80)  
   * Check regular expression patterns in attempt\_local\_calculation()
   * Investigate why "calculate the sum of 70 and 80" wasn't matched but "what is 50 + 60" was
2. **Location Parsing Bug**: In the comprehensive model, locations are split into individual characters  
   * The issue appears in create\_comprehensive\_model when handling location lists
   * Consider adding string normalization for location names
3. **API Call Optimization**: Multiple consecutive OpenAI API calls introduce latency  
   * Consider batch API requests where possible
   * Add client-side caching for repetitive categorization requests
4. **Parameter Extraction Enhancement**:  
   * Extracted "hydro" in Test 4 even though it wasn't in the prompt
   * Consider improving pattern matching or keyword extraction

My async implementation fundamentally works well and achieves true parallel processing. The issues identified are mostly related to pattern matching and text processing rather than the async architecture itself.