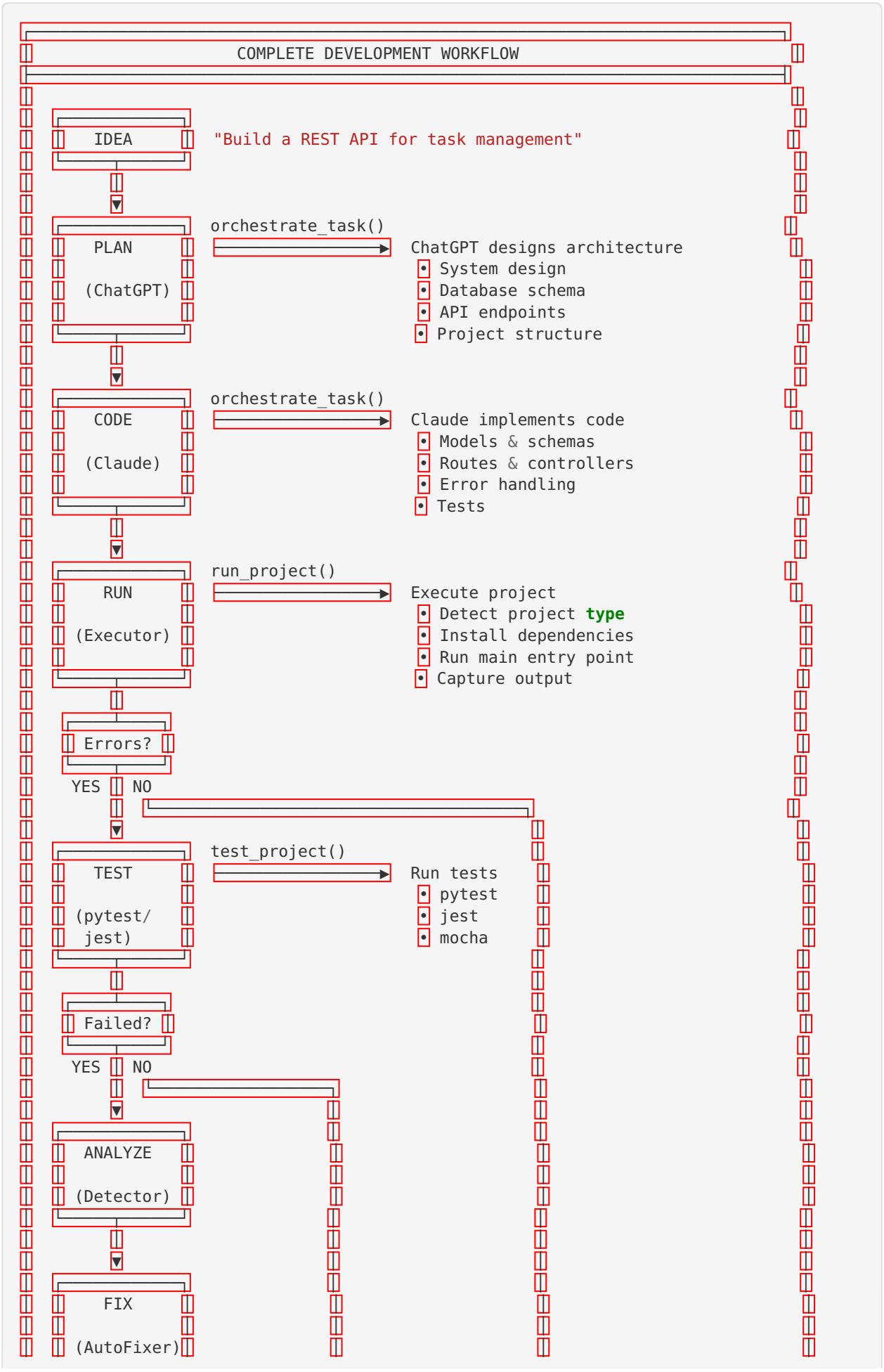
AI Orchestrator Workflows

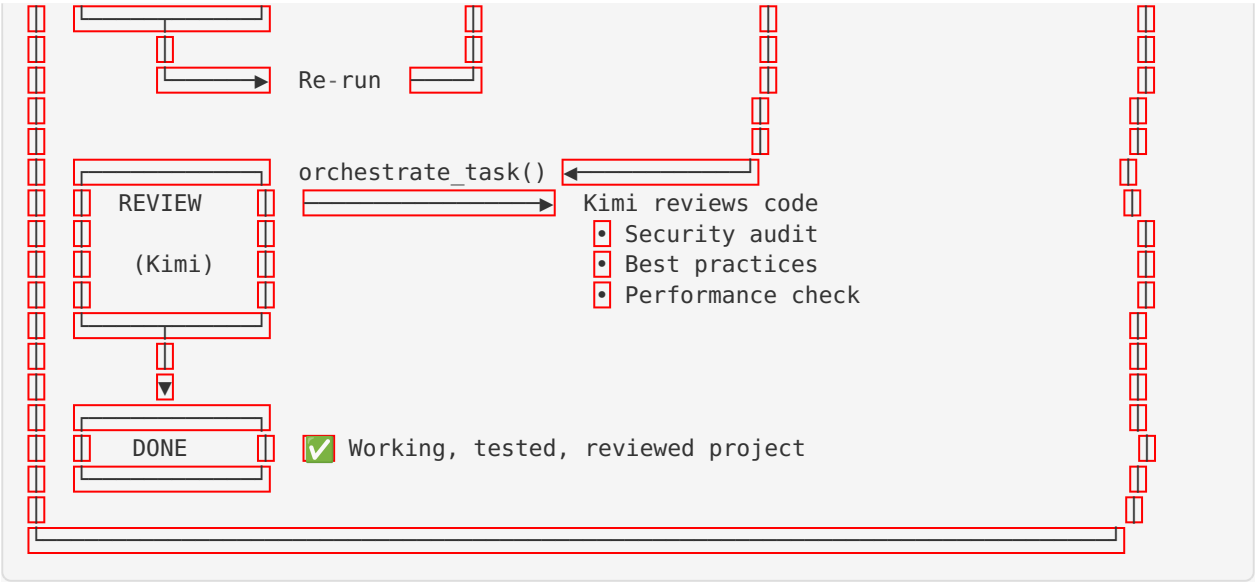
This document provides visual workflow diagrams, common development scenarios, decision trees, and integration patterns for the AI Orchestrator.

Table of Contents

Visual Workflow Diagrams

Complete Development Workflow

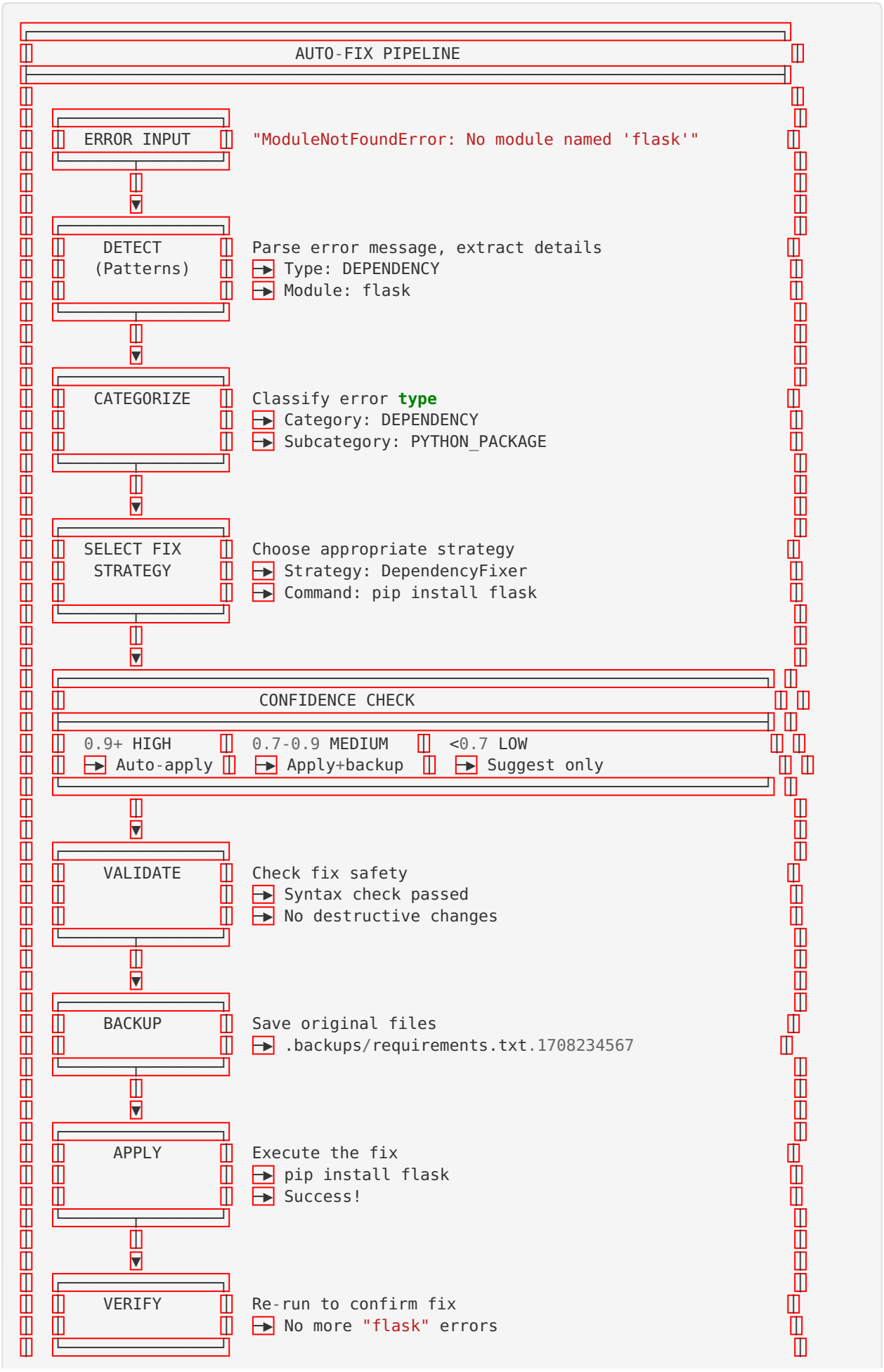

COMPLETE DEVELOPMENT WORKFLOW
IDEA
"Build a REST API for task management"
PLAN
(ChatGPT)
orchestrate_task()
ChatGPT designs architecture
System design
Database schema
API endpoints
Project structure
CODE
(Claude)
orchestrate_task()
Claude implements code
Models & schemas
Routes & controllers
Error handling
Tests
RUN
(Executor)
run_project()
Execute project
Detect project type
Install dependencies
Run main entry point
Capture output
Errors?
YES
NO
TEST
(pytest/
jest)
test_project()
Run tests
pytest
jest
mocha
Failed?
YES
NO
ANALYZE
(Detector)
FIX
(AutoFixer)



Verification Loop Detail

```
┌─────────────────────────────────────────────────────────────┐
│                    VERIFICATION LOOP                        │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│  verify_project("/path/to/project")                         │
│                                                             │
│  ┌─────────────────────────────────────────────────────┐    │
│  │                    CYCLE 1                          │    │
│  ├─────────────────────────────────────────────────────┤    │
│  │  RUN → 5 errors → FIX 3 → Progress: ████░░░░ 40%    │    │
│  └─────────────────────────────────────────────────────┘    │
│                          │                                  │
│                          ▼                                  │
│  ┌─────────────────────────────────────────────────────┐    │
│  │                    CYCLE 2                          │    │
│  ├─────────────────────────────────────────────────────┤    │
│  │  RUN → 2 errors → FIX 1 → Progress: ████████░ 80%   │    │
│  └─────────────────────────────────────────────────────┘    │
│                          │                                  │
│                          ▼                                  │
│  ┌─────────────────────────────────────────────────────┐    │
│  │                    CYCLE 3                          │    │
│  ├─────────────────────────────────────────────────────┤    │
│  │  RUN → 1 error → FIX 1 → Progress: ██████████ 100% ✅ │    │
│  └─────────────────────────────────────────────────────┘    │
│                                                             │
│  Loop Termination Conditions:                               │
│  ├─ ✅ All tests pass                                       │
│  ├─ ⚠️  Max cycles reached (default: 10)                    │
│  ├─ ⚠️  Same error 3+ times                                 │
│  └─ ⚠️  No progress in 3 cycles                             │
│                                                             │
└─────────────────────────────────────────────────────────────┘
```

Auto-Fix Pipeline


AUTO-FIX PIPELINE
ERROR INPUT
"ModuleNotFoundError: No module named 'flask'"
DETECT (Patterns)
Parse error message, extract details
Type: DEPENDENCY
Module: flask
CATEGORIZE
Classify error type
Category: DEPENDENCY
Subcategory: PYTHON_PACKAGE
SELECT FIX STRATEGY
Choose appropriate strategy
Strategy: DependencyFixer
Command: pip install flask
CONFIDENCE CHECK
0.9+ HIGH
Auto-apply
0.7-0.9 MEDIUM
Apply+backup
<0.7 LOW
Suggest only
VALIDATE
Check fix safety
Syntax check passed
No destructive changes
BACKUP
Save original files
.backups/requirements.txt.1708234567
APPLY
Execute the fix
pip install flask
Success!
VERIFY
Re-run to confirm fix
No more "flask" errors

Common Development Scenarios

Scenario 1: New Project from Scratch

```
Goal: Build a new REST API

Step 1: Design
@ai-orchestrator orchestrate_task(
  "Design a REST API for user management with:
   - User registration and login
   - JWT authentication
   - Role-based access control")

Step 2: Implement
@ai-orchestrator orchestrate_task(
  "Implement the user management API based on the design")

Step 3: Verify & Fix
@ai-orchestrator verify_project("/path/to/api")

Step 4: Review
@ai-orchestrator orchestrate_task(
  "Review the API for security and best practices")
```

Scenario 2: Debug Production Error

```
Goal: Fix a 500 error in production

Step 1: Analyze
@ai-orchestrator analyze_errors(
  "/path/to/project",
  error_log="TypeError: Cannot read property 'id' of null...")

Step 2: Understand
@ai-orchestrator route_to_model(
  "Explain why this null reference occurs and how to prevent",
  "gemini")

Step 3: Fix
@ai-orchestrator fix_issues("/path/to/project")

Step 4: Verify
@ai-orchestrator verify_project("/path/to/project")

Step 5: Review Fix Safety
@ai-orchestrator route_to_model(
  "Review this fix for production safety", "moonshot")
```

Scenario 3: Add Feature to Existing Project

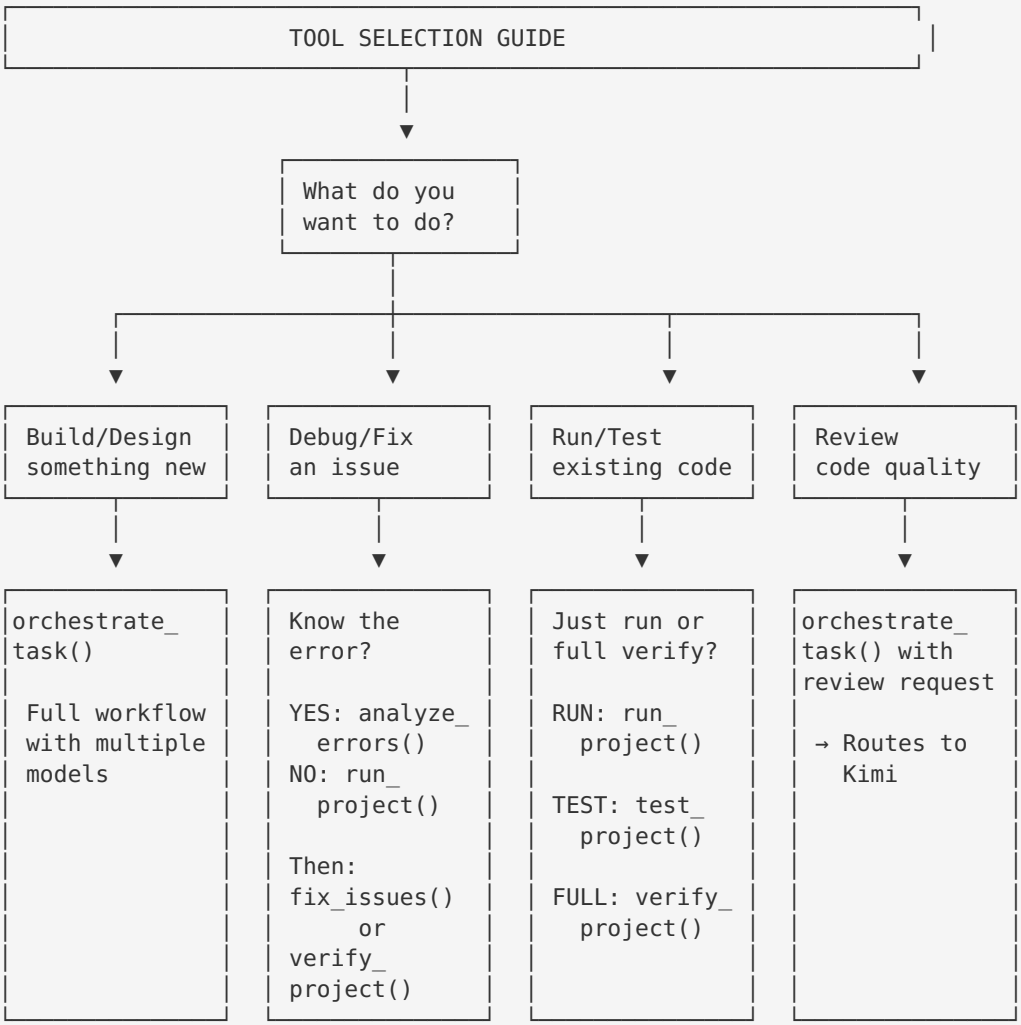
```
Goal: Add OAuth login to existing app

Step 1: Understand Existing Code
@ai-orchestrator route_to_model(
  "Analyze the auth module structure in /path/to/project",
  "gemini")

Step 2: Design Integration
@ai-orchestrator orchestrate_task(
  "Design how to add Google OAuth to the existing auth system")

Step 3: Implement
@ai-orchestrator orchestrate_task(
  "Implement Google OAuth following the integration design")

Step 4: Test
@ai-orchestrator test_project("/path/to/project")

Step 5: Full Verification
@ai-orchestrator verify_project("/path/to/project")
```

Scenario 4: Fix Failing Tests

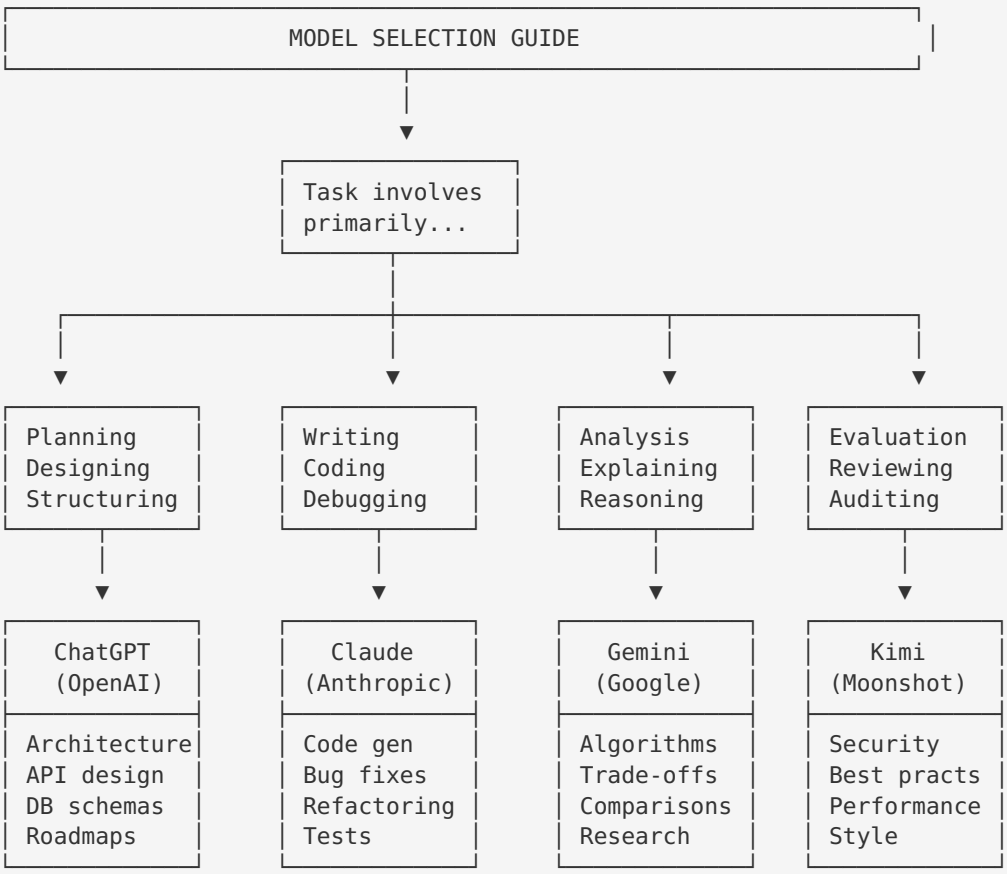
```
Goal: All tests pass

Option A: Manual Control
@ai-orchestrator test_project("/path")      # See failures
@ai-orchestrator analyze_errors("/path")    # Understand why
@ai-orchestrator fix_issues("/path")        # Apply fixes
@ai-orchestrator test_project("/path")      # Verify fixed

Option B: Automated Loop
@ai-orchestrator verify_project("/path")    # Auto fix loop
```

Decision Trees

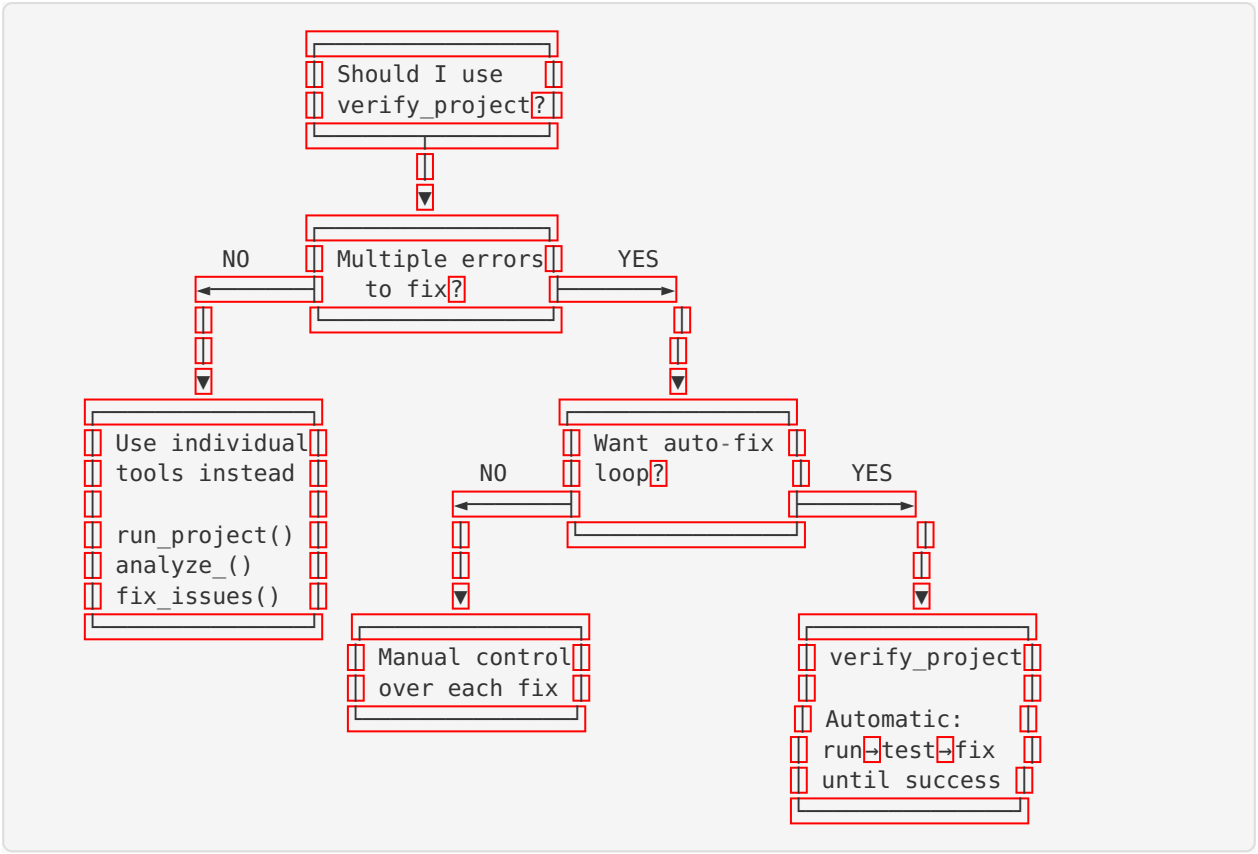
Which Tool Should I Use?



Which Model for This Task?

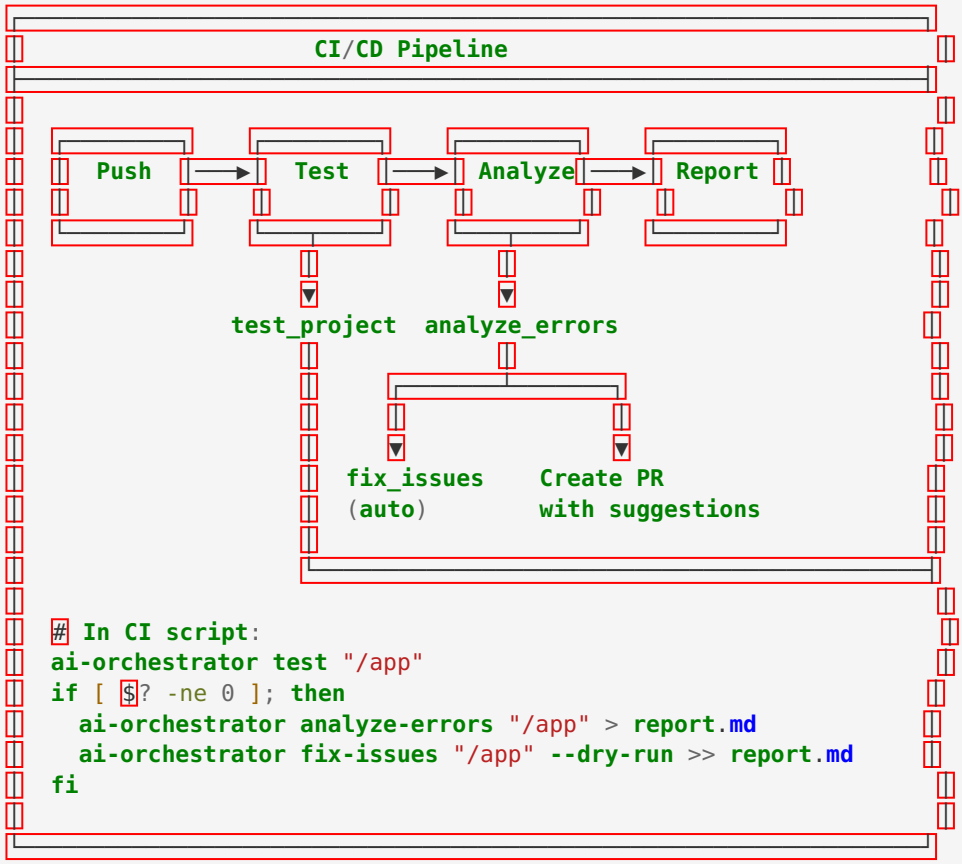


When to Use Verification Loop?



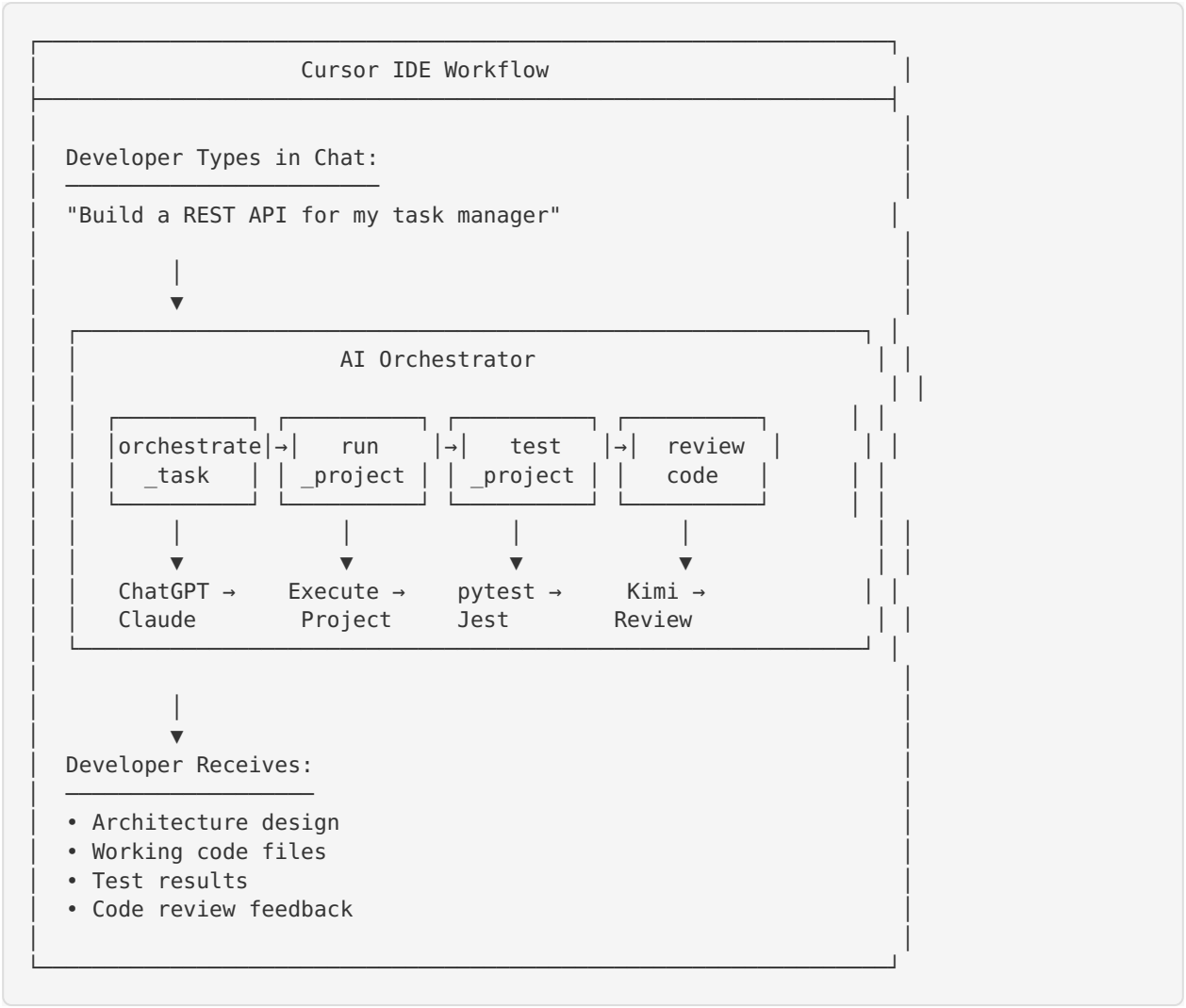
Integration Patterns

Pattern 1: CI/CD Integration




```
# In CI script:
ai-orchestrator test "/app"
if [ $? -ne 0 ]; then
  ai-orchestrator analyze-errors "/app" > report.md
  ai-orchestrator fix-issues "/app" --dry-run >> report.md
fi
```

Pattern 2: IDE Integration (Cursor)



Pattern 3: Programmatic Usage

```python
# Python Integration Example

from ai_orchestrator.orchestrator import Orchestrator
from ai_orchestrator.execution import ProjectRunner, VerificationLoop
from ai_orchestrator.config import Config

# Load configuration
config = Config.load()

# Initialize orchestrator
orchestrator = Orchestrator(config)

# Design phase
architecture = orchestrator.execute(
    "Design a REST API for user management"
)

# Implementation phase
implementation = orchestrator.execute(
    f"Implement based on this architecture: {architecture.consolidated_output}"
)

# Verification loop
loop = VerificationLoop(config)
report = loop.run_development_cycle(
    project_path="/path/to/project",
    max_cycles=10,
    run_tests=True
)

if report.status == "SUCCESS":
    # Code review
    review = orchestrator.execute(
        "Review the implementation for security and best practices",
        task_type="code_review"
    )
```

Quick Reference

Tool Summary Table

Tool	Purpose	Input	Output
`orchestrate_task`	Multi-model task	Task description	Consolidated result
`analyze_task`	Task planning	Task description	Routing plan
`route_to_model`	Direct routing	Task + model	Model response
`run_project`	Execute project	Project path	Execution result
`test_project`	Run tests	Project path	Test results
`analyze_errors`	Error analysis	Project path	Error details
`fix_issues`	Apply fixes	Project path	Fix report
`verify_project`	Full loop	Project path	Loop report

Common Commands

```
# Check available models
@ai-orchestrator check_status()

# Design something
@ai-orchestrator orchestrate_task("Design [description]")

# Build something
@ai-orchestrator orchestrate_task("Implement [feature]")

# Run a project
@ai-orchestrator run_project("/path")

# Test a project
@ai-orchestrator test_project("/path")

# Full verification
@ai-orchestrator verify_project("/path")

# Analyze specific error
@ai-orchestrator analyze_errors("/path", error_log="[error text]")
```

See Also

- DEVELOPMENT_WORKFLOW.md (cursor_integration/DEVELOPMENT_WORKFLOW.md) - Detailed workflow guide
- Examples (cursor_integration/examples/) - Real-world examples
- Sample Projects (cursor_integration/examples/sample_projects/) - Practice projects

- CURSOR_SETUP.md (cursor_integration/CURSOR_SETUP.md) - IDE setup guide