Pipeline Run

✓ COMPLETED

Requirement: Build a todo list application with add, remove, mark complete, and list tasks functionality. Include data persistence with JSON file storage, input validation, and a clean CLI interface.

RUN ID	TIMESTAMP	DURATION
71e86df3-1dad-4144-b2e2-985bdc3010b0	Feb 07, 2026 02:55 PM UTC	912.0s (15.2 min)

FRAMEWORK	PHASES
CrewAI + LiteLLM + Ollama	7 completed

Artifacts (7) Conversation Log Raw Manifest

Phase UUID: 20fc2aa7-a262-4bf9-b3d9-f0343d9f9c03 5.2 kB

1 **Requirement Analysis**

Requirement Analyst gpt-oss:20b 01_specification.md

Copy

Todo List CLI Application Specification

1. Project Title

Todo List CLI Application

2. Overview

The Todo List CLI Application is a lightweight command-line tool that allows users to manage a personal to-do list. Users can add new tasks, remove existing ones, mark tasks as complete, and view the current list of tasks. All data is persisted in a JSON file, ensuring that the task list survives across sessions. The application validates user input to prevent errors and provides a clean, intuitive command-line interface for efficient task management.

3. Functional Requirements

1. **Add Task**
   - Command: `add <task description>`

- Creates a new task with a unique ID, the provided description, and a status of *incomplete*.
- Persists the new task to the JSON file immediately.

2. **Remove Task**

Phase UUID: a21af5c8-4589-49ae-a1d5-2e1983ae233e 10.5 kB

2 Code Generation

Senior Developer qwen3-coder:30b 02_code.py Copy

```
# Phase UUID: a21af5c8-4589-49ae-a1d5-2e1983ae233e

#!/usr/bin/env python3
"""
Todo List CLI Application

A lightweight command-line tool for managing a personal to-do list.
All data is persisted in a JSON file, ensuring that the task list survives across s
"""

import json
import os
import sys
from typing import List, Dict, Optional, Any
from pathlib import Path


class TodoList:
    """
    A class to manage a todo list with persistence to a JSON file.

    This class handles adding, removing, completing, and listing tasks,
    with automatic persistence to a JSON file.
    """

    def __init__(self, filename: str = "tasks.json"):
        """
        Initialize the TodoList with a filename for persistence.

        Args:
            filename: The name of the JSON file to store tasks.
        """
        self.filename = filename
        self.tasks: List[Dict[str, Any]] = []
        self._load_tasks()

    def _load_tasks(self) -> None:
        """Load tasks from the JSON file."""
```

Phase UUID: 6da45974-2f50-45d1-8dab-8b9afd7b740e 13.8 kB

SDLC Pipeline New Run History

Code Reviewer devstral-small-2:24b 03_review.md Copy

Verdict: APPROVED

Score: 9/10

Issues Found

1. **Severity: Low**
   - **Issue**: The `list_tasks` method raises a `ValueError` for invalid filter status, but this is not handled in the main loop. This could lead to an unhandled exception.
   - **Fix**: Add a try-except block in the main loop to catch `ValueError` when calling `list_tasks`.
2. **Severity: Low**
   - **Issue**: The `get_task` method is not used consistently. In some places, the code directly iterates over `self.tasks` to find a task, while in others, it uses `get_task`.
   - **Fix**: Consistently use `get_task` for retrieving tasks by ID.
3. **Severity: Low**
   - **Issue**: The `parse_command` function does not handle empty commands gracefully. It returns an empty string and an empty list, which might not be the intended behavior.

Phase UUID: ca2ca062-a6f1-4036-9a92-ba7fd61a8929 19.0 kB

4 **Test Generation**
QA Engineer qwen3-coder:30b 04_tests.py Copy

```python
# Phase UUID: ca2ca062-a6f1-4036-9a92-ba7fd61a8929

import json
import os
import tempfile
import pytest
from unittest.mock import patch, mock_open
from pathlib import Path

# Import the module to test
import sys
import os
sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))

from todo_list import TodoList, print_help, parse_command, main


@pytest.fixture
def temp_todo_file():
    """Create a temporary file for testing."""
    with tempfile.NamedTemporaryFile(mode='w', suffix='.json', delete=False) as f:
        f.write('[]')
        temp_file = f.name

    yield temp_file

    # Clean up
```

```
    if os.path.exists(temp_file):
        os.unlink(temp_file)


@pytest.fixture
def todo_list_with_file(temp_todo_file):
    """Create a TodoList instance with a temporary file."""
    return TodoList(filename=temp_todo_file)


class TestTodoList:
```

Phase UUID: 10b7dfed-88af-432a-af4a-ca518bc3d599 12.5 kB

5 **Documentation**
Tech Writer gpt-oss:20b 05_documentation.md Copy

README.md

```
# Todo List CLI Application

A lightweight, cross-platform command-line tool for managing a persona
All data is persisted in a JSON file (`tasks.json`) so your tasks surv

> **Why this tool?**
> • No external dependencies – just Python 3.8+
> • Atomic writes for data safety
> • Simple, intuitive commands
> • Extensible – add due dates, priorities, etc. in the future

---

## Table of Contents

- [Installation](#installation)
- [Usage](#usage)
- [API Reference](#api-reference)
- [Configuration](#configuration)
- [Troubleshooting](#troubleshooting)
- [Contributing](#contributing)
- [License](#license)

---

## Installation

The application is a single Python script.
No installation is required beyond having Python 3.8+ installed
```

Phase UUID: c14a3edc-76cb-4f86-ba3a-d54d5001cb67 3.6 kB

6 **Devops Configuration**
DevOps Agent gpt-oss:20b 06_devops.md Copy

Dockerfile

```
# syntax=docker/dockerfile:1.4
# ---- Build stage ----
FROM python:3.11-slim AS builder
WORKDIR /app
COPY todo_cli.py .
# No external dependencies, but keep pip updated for future use
RUN python -m pip install --no-cache-dir --upgrade pip

# ---- Runtime stage ----
FROM python:3.11-slim AS runtime
WORKDIR /app
COPY --from=builder /app /app

# Create a non-root user
RUN addgroup --system appgroup && \
    adduser --system --ingroup appgroup appuser

USER appuser

# Health check - simply verifies that the interpreter can start
HEALTHCHECK --interval=30s --timeout=5s --start-period=5s --retries=3
  CMD python -c "import sys; sys.exit(0)" || exit 1

ENTRYPOINT ["python", "todo_cli.py"]
```

docker-compose.yml

Phase UUID: 71428240-de57-4c5e-b92d-db44e0560d6f 11.3 kB

7 **Ui Design**

UI Designer gpt-oss:20b 07_ui_app.py

Copy

```
# Phase UUID: 71428240-de57-4c5e-b92d-db44e0560d6f

#!/usr/bin/env python3
"""
Streamlit UI for the Todo List CLI Application
"""

import json
import os
from pathlib import Path
from typing import Any, Dict, List, Optional

import streamlit as st


# ---------------------------------------------------------------------- #
#  Persistence layer - same logic as the CLI version
# ---------------------------------------------------------------------- #
class TodoList:
    """
    Manage a todo list with persistence to a JSON file.
    """

    def __init__(self, filename: str = "tasks.json"):
        self.filename = Path(filename)
```

```
        self.tasks: List[Dict[str, Any]] = []
        self._load_tasks()

    def _load_tasks(self) -> None:
        """Load tasks from the JSON file."""
        if self.filename.exists():
            try:
                with self.filename.open("r", encoding="utf-8") as f:
                    self.tasks = json.load(f)
            except (json.JSONDecodeError, OSError) as exc:
                st.exception(f"Failed to load tasks: {exc}")
                self.tasks = []
        else:
```

Expand All Collapse All

+ New Pipeline Back to History