Verification Tool - Step-by-Step Execution Guide

Project Description

The tool performs three critical validation checks: file structure verification, database schema validation, and API endpoint testing. It generates tamper-proof evidence bundles with cryptographic hashes to ensure verification results cannot be falsified.

Step-by-Step Execution Guide

Step 1: Install Dependencies

```
# pip install requests
```

Step 2: Create Test Files

```
# python setup_test_environment.py
```

Step 3: Run Basic Verification

```
# python verify.py --phase 2
```

Step 4: Check Evidence Created

```
# ls evidence/phase2/
```

Step 5: Test Simulated Failure

```
# python verify.py --phase 2 --simulate-fail file_structure
```

Step 6: Test All Failure Types

```
# python verify.py --phase 2 --simulate-fail database_schema

# python verify.py --phase 2 --simulate-fail api_endpoint
```

Step 7: Test Real Failure

```
# mv test_files/config.json test_files/config.json.bak

# python verify.py --phase 2

# mv test_files/config.json.bak test_files/config.json
```

Step 8: Verify Evidence Integrity

```
# cat evidence/phase2/[latest-timestamp]/manifest.json
```

Expected Results Summary

Success Criteria:

- Step 3: Exit code 0, Status: PASS, all checks show ✅
- Step 4: Timestamped evidence folder with report.json, verify.log, manifest.json, and artifacts/
- Steps 5-6: Exit code 1, Status: FAIL, specific check shows ❌
- Step 7: File structure check fails with missing file error
- Step 8: JSON file with SHA256 hashes proving tamper resistance

Tool Validation Complete: All steps demonstrate proper automated verification with evidence generation suitable for milestone-based AI system development.