# Unit Tests for Summarizer Project

This directory contains unit tests for the Summarizer 1.1 project.

## Test Files

- **test_helper.py** - Tests for utility helper functions (percentage calculation, comparison logic, embedding parsing)
- **test_employee.py** - Tests for employee analytics utilities (deeds calculation, statement generation)
- **test_knowledge_generator.py** - Tests for the KnowledgeGenerator service
- **test_report_generator.py** - Tests for the ReportGenerator service

## Running Tests

### Run all tests:

```
python -m unittest discover -s tests -p "test_*.py" -v
```

### Run a specific test file:

```
python -m unittest tests.test_helper -v
python -m unittest tests.test_employee -v
python -m unittest tests.test_knowledge_generator -v
python -m unittest tests.test_report_generator -v
```

### Run a specific test class:

```
python -m unittest tests.test_helper.TestHelperFunctions -v
```

### Run a specific test method:

```
python -m unittest
tests.test_helper.TestHelperFunctions.test_percentage_basic -v
```

## Dependencies

The test suite uses Python's built-in `unittest` module and the `unittest.mock` library. Make sure you have the following installed:

- Python 3.7+
- unittest (built-in)
- unittest.mock (built-in)
- numpy (for report generator tests)

## Test Coverage

### Helper Tests (test_helper.py)

- ✅ Percentage calculations with various inputs
- ✅ Comparison logic (above/below/at average)
- ✅ Embedding parsing from different formats (list, string, Postgres format)
- ✅ Error handling for invalid embeddings

### Employee Tests (test_employee.py)

- ✅ Single employee deeds extraction and calculation
- ✅ Breakdown, pause, and production metrics
- ✅ Average comparison calculations
- ✅ Employee statement generation
- ✅ Handling non-existent employees

### Knowledge Generator Tests (test_knowledge_generator.py)

- ✅ Initialization with storage and embedding model
- ✅ Handling factories with no logs
- ✅ Processing factories with actual logs
- ✅ Error handling during encoding
- ✅ Statement generation for employees and machines

### Report Generator Tests (test_report_generator.py)

- ✅ Initialization and setup
- ✅ Query validation
- ✅ Vector handling and similarity calculations
- ✅ Report generation with valid records
- ✅ Top-K filtering
- ✅ System prompt formatting for LLM

## Notes

- Tests use mocking extensively to avoid dependencies on external services (Supabase, Qwen3 model)
- Each test is independent and can run in any order
- All external service calls are mocked for fast, reliable test execution