

# Phase 3 Report

Mar 2025

Bingchen Yang

# Table of Contents

Table of Contents.....	2
1. Test Plan.....	3
1.1 Introduction.....	3
1.2. Objectives.....	3
1.3. Scope.....	3
1.4. Testing Strategy.....	3
1.4.1 Unit Tests.....	3
1.4.2 Integration Tests.....	4
1.4.3 System Tests.....	4
1.5. Test Execution.....	5
1.6. Test Report.....	6
2. Test Report.....	7
2.1 Test Results.....	7
2.1.1 Unit Tests.....	7
2.1.2 Integration Tests.....	8
2.1.3 System Tests.....	8
2.2 Bug Fixes.....	8
2.2.1 Bug Identified:.....	8
2.2.2 Proposed Fixes:.....	9
2.3 Iteration Logs.....	9
Iteration Logs.....	9
3. Individual Contributions.....	11
Bingchen Yang.....	11
4. Team Leader's Role.....	12
5. Reference.....	13

# 1. Test Plan

## 1.1 Introduction

This test plan outlines the testing strategy for the `InsightifyExtractor` class and the main module in the project. The testing strategy includes unit, integration, and system tests to ensure the functionality and reliability of the project.

## 1.2. Objectives

The objectives of this test plan are:

- To validate the functionality of individual units (unit tests).
- To verify the interaction between integrated units (integration tests).
- To ensure the overall system works as expected (system tests).

## 1.3. Scope

The scope of this test plan includes:

- Testing the `InsightifyExtractor` class.
- Testing the main module functions (`clear_output_directory`, `select_file`, and main logic).

## 1.4. Testing Strategy

### 1.4.1 Unit Tests

Unit tests focus on testing individual functions in isolation. The following functions will be tested:

- `clear_output_directory`
- `select_file`
- `InsightifyExtractor.__init__`
- `InsightifyExtractor.extract_text_from_page`
- `InsightifyExtractor.extract_images_from_page`
- `InsightifyExtractor.extract_tables_structured_from_page`
- `InsightifyExtractor.load_and_extract_content`
- `InsightifyExtractor.dump_to_markdown`

Unit Test Cases

1. Test Case: `main.clear_output_directory`

- Verify that the output directory is cleared correctly.
  - Expected outcome: The directory is empty after calling the function.
- 2. Test Case: `main.select_file`
  - Verify that the file selection dialog works correctly and returns the expected file path.
  - Expected outcome: The returned file path matches the mock value.
- 3. Test Case: `InsightifyExtractor.extract_text_from_page`
  - Verify that text extraction from a page excludes content from tables correctly.
  - Expected outcome: The extracted text does not contain table content.
- 4. Test Case: `InsightifyExtractor.extract_images_from_page`
  - Verify that images are extracted correctly from a page.
  - Expected outcome: The extracted image paths exist.
- 5. Test Case: `InsightifyExtractor.extract_tables_structured_from_page`
  - Verify that structured tables are extracted correctly from a page.
  - Expected outcome: The extracted table paths exist and contain the expected content.
- 6. Test Case: `InsightifyExtractor.load_and_extract_content`
  - Verify that content is loaded and extracted correctly from multiple PDFs.
  - Expected outcome: The extracted content matches the expected content.
- 7. Test Case: `InsightifyExtractor.dump_to_markdown`
  - Verify that the content is dumped to a Markdown file correctly.
  - Expected outcome: The dumped Markdown file exists and contains the expected content.

## 1.4.2 Integration Tests

Integration tests focus on testing the interaction between integrated units. The following scenarios will be tested:

- Interaction between `clear_output_directory`, `select_file`, and `InsightifyExtractor` in the main logic.

### Integration Test Cases

1. Test Case: Main Logic
  - Verify the interaction between `clear_output_directory`, `select_file`, and `InsightifyExtractor`.
  - Expected outcome: The output directory is cleared, the file is selected, and the content is extracted and dumped correctly.

## 1.4.3 System Tests

System tests focus on testing the overall system functionality. The following end-to-end scenarios will be tested:

- End-to-end extraction and conversion of a PDF to Markdown format.

### System Test Cases

### 1. Test Case: End-to-End PDF to Markdown Conversion

- Verify the entire process of extracting content from a PDF and converting it to Markdown format.
- Expected outcome: The content is extracted from the PDF and dumped to a Markdown file correctly.

## 1.5. Test Execution

The tests will be executed using the `pytest` framework with the following command:

configuration

The screenshot shows a configuration window for a test runner. At the top, there is a 'Name' field containing 'pytest in test\_main.py', an unchecked 'Allow multiple instances' checkbox, and a checked 'Store as project file' checkbox with a gear icon. Below this is a tabbed interface with 'Configuration' and 'Logs' tabs. The 'Configuration' tab is active and contains the following fields: 'Target' with radio buttons for 'Module name', 'Script path' (selected), and 'Custom'; a text box for the script path containing 'D:/AlgomaU/COCS5506/Group\_F\_Project/Insightify/tests/test\_main.py'; 'Keywords' and 'Parameters' text boxes; 'Additional Arguments' with the value '-html=test\_main\_report.html --self-contained-html --capture=tee-sys -v'; an expanded 'Environment' section with 'Environment variables' text box; 'Python interpreter' set to 'Python 3.9 C:\Users\13079\AppData\Local\Programs\Python\Python39\'; 'Interpreter options' text box; and 'Working directory' text box.

Name:  ☐ Allow multiple instances ☒ Store as project file

Configuration Logs

pytest

Target: ☐ Module name ☒ Script path ☐ Custom

Keywords

Parameters

Additional Arguments


▼ Environment

Environment variables:

Python interpreter: Python 3.9 C:\Users\13079\AppData\Local\Programs\Python\Python39\ ▼

Interpreter options:


Working directory:

Name:  ☐ Allow multiple instances ☒ Store as project file 

Configuration Logs

pytest

Target: ☐ Module name ☒ Script path ☐ Custom




Keywords



Parameters


Additional Arguments


▼ Environment

Environment variables:



Python interpreter:  Python 3.9 C:\Users\13079\AppData\Local\Programs\Python\Python39\ 

Interpreter options:  

Working directory:  

## 1.6. Test Report

Detailed test reports will be generated in HTML format and will include the results of all unit and integration tests. System test will generate a markdown file to contrast with the original PDF file.

## 2. Test Report

### 2.1 Test Results

#### [test\\_extractor.html](#)


##### Summary

5 tests took 00:00:15.

(Un)check the boxes to filter the results.

☒ 4 Failed, ☒ 1 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☐ 0 Reruns

[Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Passed	test_insightify_extractor.py::test_extract_images_from_page	337 ms	
Failed	test_insightify_extractor.py::test_dump_to_markdown	00:00:05	
Failed	test_insightify_extractor.py::test_load_and_extract_content	00:00:08	
Failed	test_insightify_extractor.py::test_extract_tables_structured_from_page	833 ms	
Failed	test_insightify_extractor.py::test_extract_text_from_page	848 ms	

#### [test\\_main.html](#)


##### Summary

3 tests took 144 ms.

(Un)check the boxes to filter the results.

☒ 1 Failed, ☒ 2 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☐ 0 Reruns

[Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Failed	test_main.py::test_main_logic	72 ms	
Passed	test_main.py::test_clear_output_directory	4 ms	
Passed	test_main.py::test_select_file	68 ms	

### 2.1.1 Unit Tests

1. Test Case: `clear_output_directory`
  - Outcome: Passed
  - Details: Verified that the output directory is cleared correctly.
2. Test Case: `select_file`
  - Outcome: Passed
  - Details: Verified that the file selection dialog works correctly and returns the expected file path.
3. Test Case: `InsightifyExtractor.extract_text_from_page`
  - Outcome: Failed
  - Details: The test case failed due to an `AssertionError`. The error indicates that the extracted text contains table content. The expected text was "This is a test text.", but the extracted text included table data as well.
4. Test Case: `InsightifyExtractor.extract_images_from_page`
  - Outcome: Passed
  - Details: Verified that images are extracted correctly from a page.
5. Test Case: `InsightifyExtractor.extract_tables_structured_from_page`
  - Outcome: Failed

- Details: The test case failed due to an `AssertionError`. The error indicates that the length of `table_paths` is `0` instead of the expected `1`. This error occurred because the model type used is not supported for all configurations.
- 6. Test Case: `InsightifyExtractor.load_and_extract_content`
  - Outcome: Failed
  - Details: The test case failed due to a `TypeError`. The error indicates that `NoneType` object is not iterable. This occurred because `table_paths` was `None` during the execution of `load_table_content`.
- 7. Test Case: `InsightifyExtractor.dump_to_markdown`
  - Outcome: Failed
  - Details: The test case failed due to a `TypeError`. The error indicates that `NoneType` object is not iterable. This occurred because `table_paths` was `None` during the execution of `load_table_content`.

## 2.1.2 Integration Tests

1. Test Case: Main Logic
  - Outcome: Failed
  - Details: The integration test case failed due to a `ValueError`. The error indicates that the directory `/path/to` does not exist. This error occurred because the mocked file path does not correspond to an actual directory.

## 2.1.3 System Tests

Original PDF: [test\\_pdf](#)

Output markdown: [output\\_md](#)

Outcome: Passed

Details: Even some unit/integration tests failed in the process, the system tests passed with decent competence on extracting content from `test_pdf` file and successfully generated a readable markdown with all details.

## 2.2 Bug Fixes

### 2.2.1 Bug Identified:

- `ValueError`: Directory `/path/to` does not exist. in the `test_main_logic`.
- `AssertionError`: `assert 'This is a test text. col1,col2,col3 1,2,3 4,5,6 7,8,9' == 'This is a test text.'` in the `test_extract_text_from_page`.
- `AssertionError`: `assert 0 == 1` in the `test_extract_tables_structured_from_page`.
- `TypeError`: `'NoneType' object is not iterable` in the `test_load_and_extract_content` and `test_dump_to_markdown`.



## 2.2.2 Proposed Fixes:

- Ensure the mocked file path corresponds to a valid temporary directory created during the test execution for `test_main_logic`.
- Investigate why the table content is not excluded correctly from the extracted text in `test_extract_text_from_page`.
- Investigate why the model type used is not supported for all configurations in `test_extract_tables_structured_from_page`.
- Investigate why `table_paths` is `None` in `load_table_content` during the `test_load_and_extract_content` and `test_dump_to_markdown` and ensure it receives the correct value.

## 2.3 Iteration Logs

### Iteration Logs

#### Iteration 1: Initial Setup

- Date: February 9, 2025
- Activities:
  - Discussed the initial setup of the `InsightifyExtractor` class and the main module (`main.py`).
  - Created the initial versions of `insightify_extractor.py` and `main.py` files.
- Changes Made: Created initial versions of the files.
- Issues Encountered: N/A
- Resolutions: N/A
- Results: Successfully created initial versions of the files.
- Next Steps: Start writing unit tests for core functions.

#### Iteration 2: Writing Unit Tests

- Date: March 5, 2025
- Activities:
  - Created initial unit tests for functions in `insightify_extractor.py` and `main.py`.
  - Discussed the test cases for core functions.
- Changes Made: Added unit test cases in `test_insightify_extractor.py` and `test_main.py`.
- Issues Encountered:
  - Mocking issues with `InsightifyExtractor`.
  - Challenges in handling file encoding in `load_table_content`.
- Resolutions:
  - Adjusted test cases to mock `InsightifyExtractor`.
  - Specified file encoding to resolve encoding issues.
- Results: All initial unit test cases passed.
- Next Steps: Proceed to integration testing.

### Iteration 3: Integration Testing

- Date: March 6, 2025
- Activities:
  - Conducted integration tests for the main logic involving `clear_output_directory`, `select_file`, and `InsightifyExtractor`.
  - Verified interactions between these components.
- Changes Made: Conducted integration tests.
- Issues Encountered:
  - Encountered `ValueError` due to invalid mocked directory path.
- Resolutions:
  - Investigated the issue and planned to adjust the mocked paths.
- Results: Integration tests revealed an issue with the mocked directory path.
- Next Steps: Adjust mocked directory paths and re-test.

### Iteration 4: Debugging and Fixing Issues

- Date: March 7, 2025
- Activities:
  - Investigated and debugged issues related to mocked directory paths.
  - Analyzed the causes of test failures in unit tests for text extraction, table extraction, and markdown dumping.
- Changes Made: N/A
- Issues Encountered:
  - Mocked directory path issues in integration tests.
  - Text extraction including table content in unit tests.
  - Table paths being `None` in `load_table_content`.
- Resolutions:
  - Investigated the issues and proposed adjustments.
- Results: Identified areas needing adjustment in tests.
- Next Steps: Make necessary adjustments and re-run tests.

### Iteration 5: Final Validation

- Date: March 10, 2025
- Activities:
  - Re-ran all unit and integration tests to ensure stability.
  - Validated all components and their interactions.
- Changes Made: N/A
- Issues Encountered:
  - Continued challenges with integration tests and certain unit tests.
- Resolutions:
  - Planned to make final adjustments based on test results.
- Results: Noted the need for further investigation and adjustments.
- Next Steps: Finalize the test report and propose fixes for identified issues.

# 3. Individual Contributions

## Bingchen Yang

### Coding and Documenting the `InsightifyExtractor` Module

- Responsibilities: Developed the `InsightifyExtractor` class, which includes methods for extracting text, images, and tables from PDF documents. Documented the code to ensure clarity and maintainability.
- Rationale for Design Decisions:
  - Choice of Libraries: Chose `fitz` (PyMuPDF) for PDF processing due to its efficiency in handling large documents and extracting various content types.
  - Modular Approach: Designed the class with modular methods (`extract_text_from_page`, `extract_images_from_page`, `extract_tables_structured_from_page`) to isolate functionality and simplify testing.
  - Error Handling: Implemented robust error handling to manage exceptions gracefully, ensuring the extractor can handle a variety of PDF formats and content structures.

### Coding and Documenting the Main Module

- Responsibilities: Developed the main module (`main.py`), which orchestrates the entire extraction process. Integrated the `InsightifyExtractor` with file selection and output management functionalities.
- Rationale for Design Decisions:
  - User Interaction: Included functions like `select_file` for an intuitive file selection process, enhancing the user experience.
  - Output Management: Implemented `clear_output_directory` to ensure a clean slate for each extraction operation, preventing old data from contaminating new results.
  - Environment Variables: Utilized the `load_dotenv` function to securely manage API keys and other sensitive configurations.

### Writing Unit and Integration Tests

- Responsibilities: Focused on developing unit and integration tests for both the `InsightifyExtractor` and the main module. Ensured the reliability and correctness of the code through comprehensive testing.
- Rationale for Design Decisions:
  - Coverage: Aimed for high test coverage to catch potential issues early and ensure all functionalities are thoroughly tested.
  - Mocking: Used mocking techniques to simulate external dependencies (e.g., file selection dialogs, API calls) for isolated testing.
  - Thorough Testing: Ensured that both unit and integration tests are comprehensive to validate the interactions between different components.

## 4. Team Leader's Role

Monitoring Progress: Tracked the progress of each module to ensure timely completion. Regularly reviewed code and documentation to maintain high standards.

Resolving Design Conflicts: Facilitated discussions to reach a consensus on design conflicts. Ensured that design decisions aligned with the project's goals and requirements.

## 5. Reference

[GitHub Repository](#)  
[test\\_files](#)