INTELLIGENT AGENTS

Document Forensics

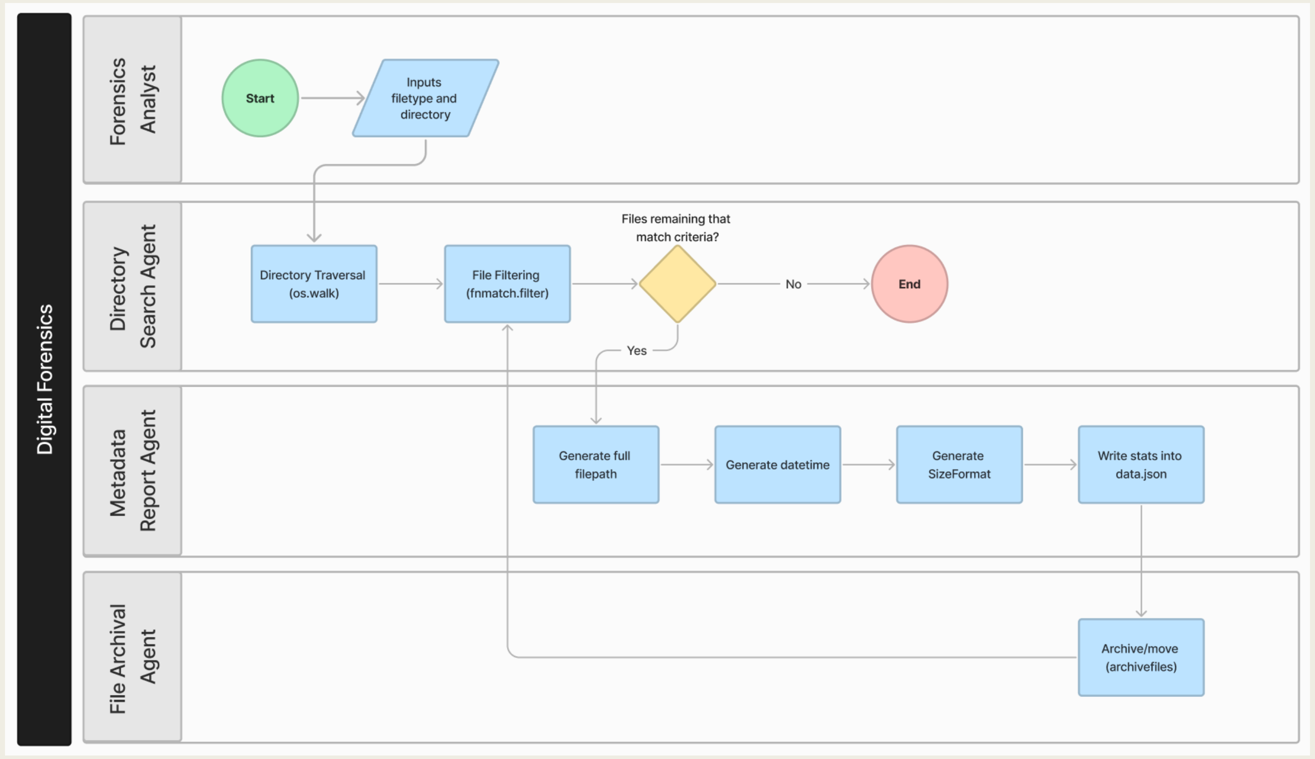
Agenda

- Why Digital Forensics?
- Software Version Comparison
  - *Without AI infusion (v1)*
  - *With AI infusion (v2)*
- Key Functions/Classes per Agent
- Unit + Integration Testing
- Functional Demo (Harmless files)
- Functional Demo (Suspicious file)
- Reference List

Why Digital Forensics?

The EU estimates that **85%** of all criminal investigations involve Electronic Evidence (Casino et al., 2022)

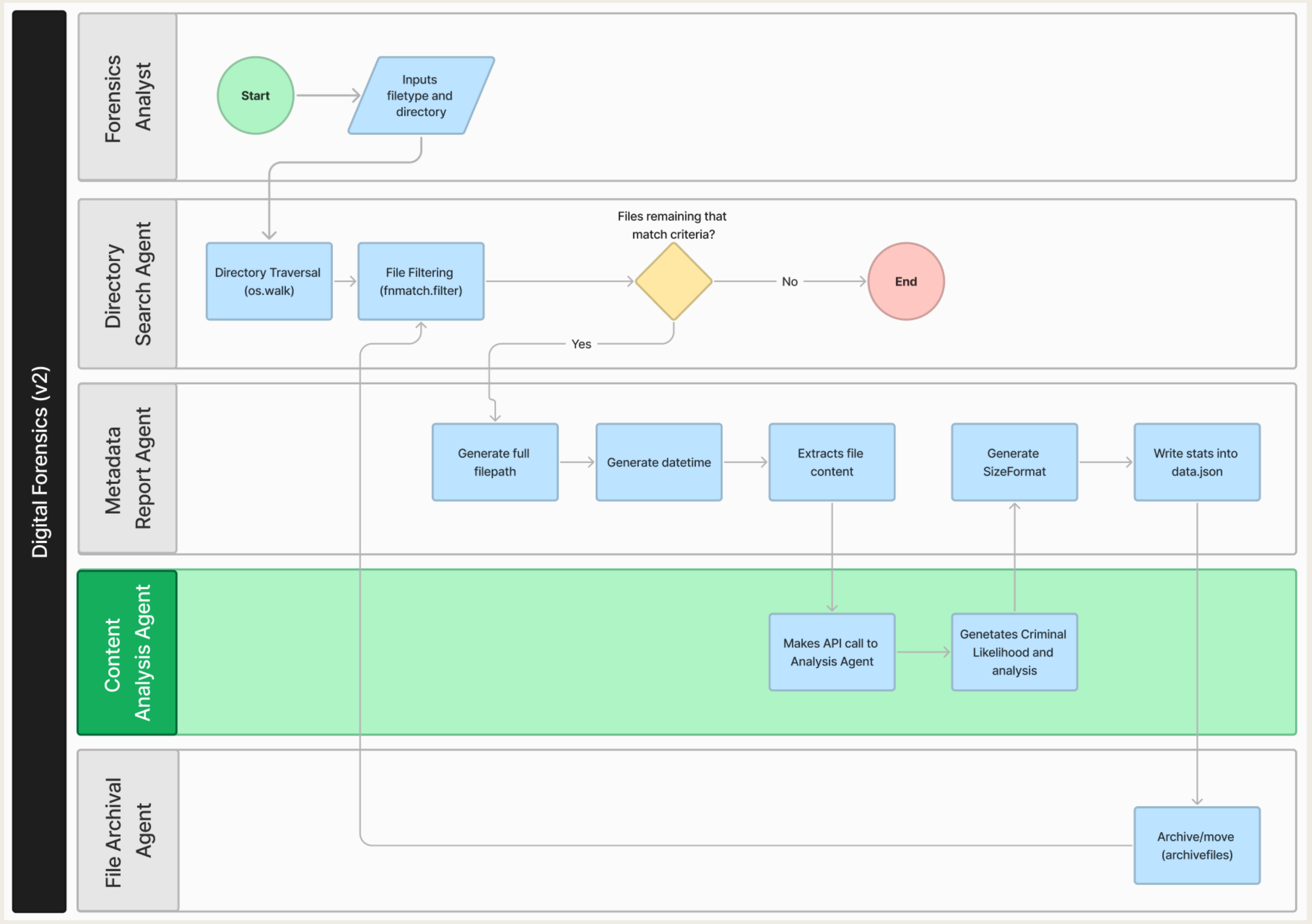
Software Version Comparison (V1)


Digital Forensics
Forensics Analyst
Directory Search Agent
Metadata Report Agent
File Archival Agent
Start
Inputs filetype and directory
Directory Traversal (os.walk)
File Filtering (fnmatch.filter)
Files remaining that match criteria?
No
End
Yes
Generate full filepath
Generate datetime
Generate SizeFormat
Write stats into data.json
Archive/move (archivefiles)

V1 (9/9/2024):

- Composed of 3 main agents:
  - *Directory Search*
  - *Medata Report*
  - *File Archival*
- No AI infusion in the process

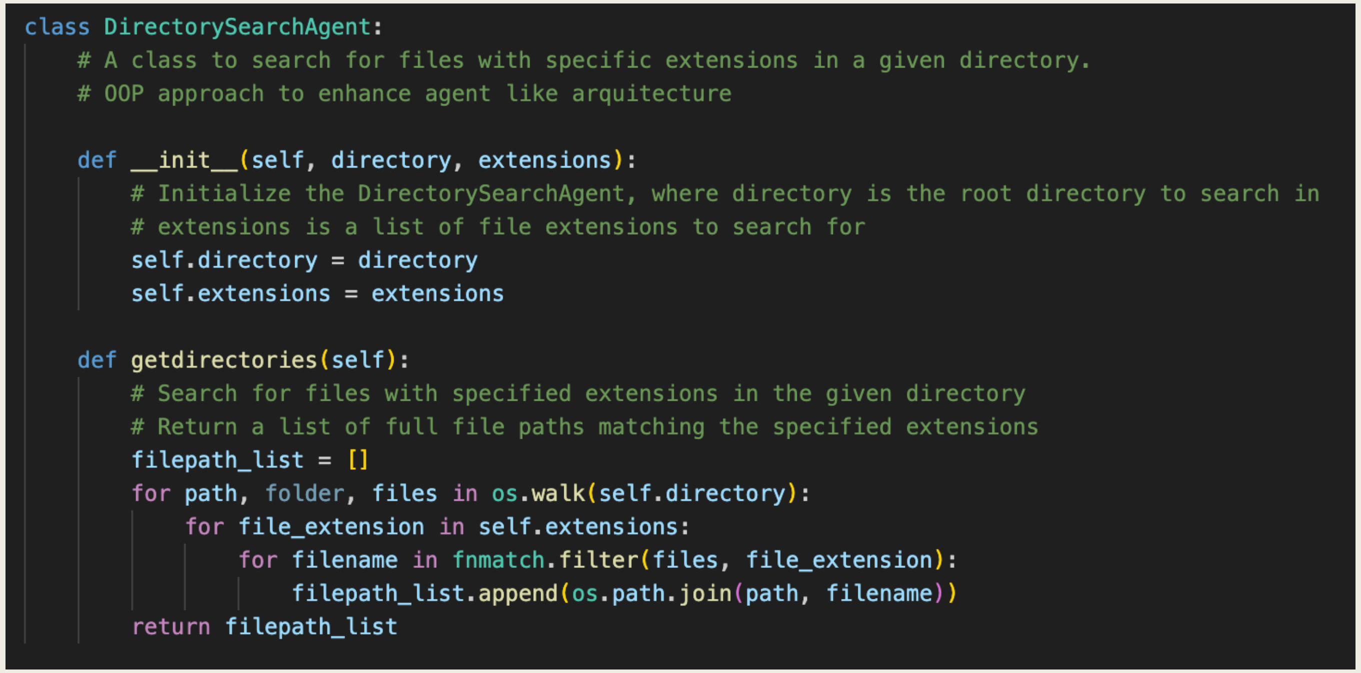
Software Version Comparison (V2)


Digital Forensics (v2)
Forensics Analyst
Directory Search Agent
Metadata Report Agent
Content Analysis Agent
File Archival Agent
Start
Inputs filetype and directory
Directory Traversal (os.walk)
File Filtering (fnmatch.filter)
Files remaining that match criteria?
No
End
Yes
Generate full filepath
Generate datetime
Extracts file content
Generate SizeFormat
Write stats into data.json
Makes API call to Analysis Agent
Genetates Criminal Likelihood and analysis
Archive/move (archivefiles)

V2 (13/10/2024):

- Added a new AI Agent to the workflow: Content Analysis
- Improved json output
- General improvements
- 93% Global Unit Test coverage
- Integration Tests

Directory Search Agent



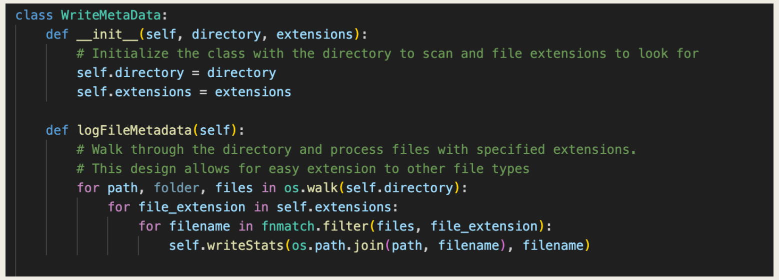
```python
class DirectorySearchAgent:
    # A class to search for files with specific extensions in a given directory.
    # OOP approach to enhance agent like arquitecture

    def __init__(self, directory, extensions):
        # Initialize the DirectorySearchAgent, where directory is the root directory to search in
        # extensions is a list of file extensions to search for
        self.directory = directory
        self.extensions = extensions

    def getdirectories(self):
        # Search for files with specified extensions in the given directory
        # Return a list of full file paths matching the specified extensions
        filepath_list = []
        for path, folder, files in os.walk(self.directory):
            for file_extension in self.extensions:
                for filename in fnmatch.filter(files, file_extension):
                    filepath_list.append(os.path.join(path, filename))
        return filepath_list
```

Libraries:

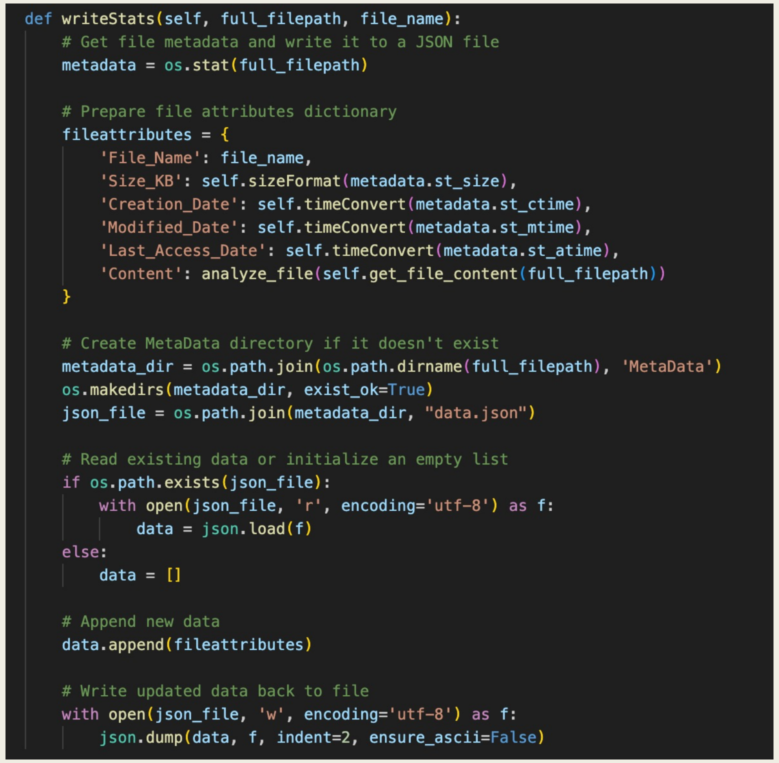
- Os
- Fnmatch

Key function: getdirectories

Metadata Report Agent



```python
class WriteMetaData:
    def __init__(self, directory, extensions):
        # Initialize the class with the directory to scan and file extensions to look for
        self.directory = directory
        self.extensions = extensions

    def logFileMetadata(self):
        # Walk through the directory and process files with specified extensions.
        # This design allows for easy extension to other file types
        for path, folder, files in os.walk(self.directory):
            for file_extension in self.extensions:
                for filename in fnmatch.filter(files, file_extension):
                    self.writeStats(os.path.join(path, filename), filename)
```



```python
def writeStats(self, full_filepath, file_name):
    # Get file metadata and write it to a JSON file
    metadata = os.stat(full_filepath)

    # Prepare file attributes dictionary
    fileattributes = {
        'File_Name': file_name,
        'Size_KB': self.sizeFormat(metadata.st_size),
        'Creation_Date': self.timeConvert(metadata.st_ctime),
        'Modified_Date': self.timeConvert(metadata.st_mtime),
        'Last_Access_Date': self.timeConvert(metadata.st_atime),
        'Content': analyze_file(self.get_file_content(full_filepath))
    }

    # Create MetaData directory if it doesn't exist
    metadata_dir = os.path.join(os.path.dirname(full_filepath), 'MetaData')
    os.makedirs(metadata_dir, exist_ok=True)
    json_file = os.path.join(metadata_dir, "data.json")

    # Read existing data or initialize an empty list
    if os.path.exists(json_file):
        with open(json_file, 'r', encoding='utf-8') as f:
            data = json.load(f)
    else:
        data = []

    # Append new data
    data.append(fileattributes)

    # Write updated data back to file
    with open(json_file, 'w', encoding='utf-8') as f:
        json.dump(data, f, indent=2, ensure_ascii=False)
```

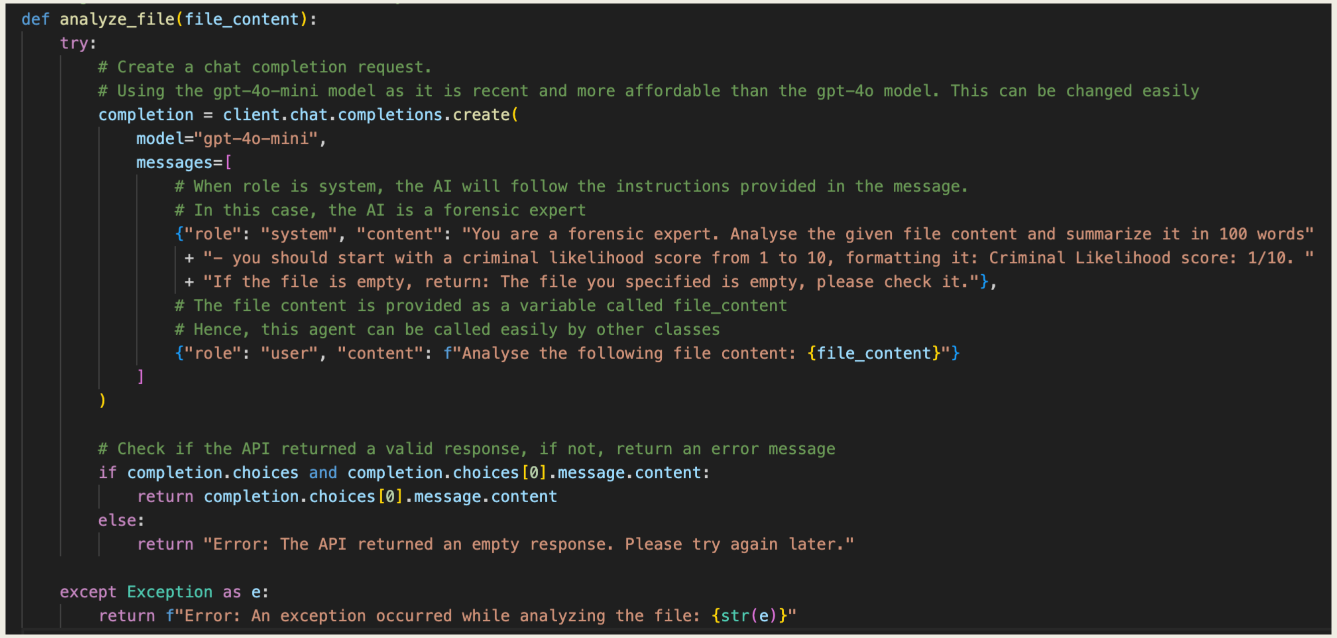
Libraries:

- Os
- Fnmatch
- JSON

Key functions:

- logFileMetadata
- writeStats

Content Analysis Agent



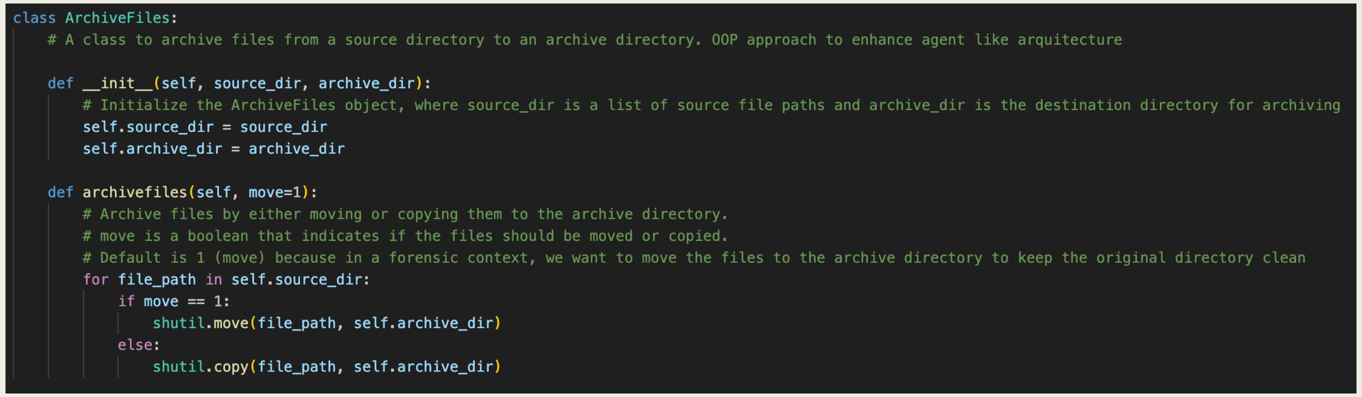
```python
def analyze_file(file_content):
    try:
        # Create a chat completion request.
        # Using the gpt-4o-mini model as it is recent and more affordable than the gpt-4o model. This can be changed easily
        completion = client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[
                # When role is system, the AI will follow the instructions provided in the message.
                # In this case, the AI is a forensic expert
                {"role": "system", "content": "You are a forensic expert. Analyse the given file content and summarize it in 100 words"
                 + "- you should start with a criminal likelihood score from 1 to 10, formatting it: Criminal Likelihood score: 1/10. "
                 + "If the file is empty, return: The file you specified is empty, please check it."},
                # The file content is provided as a variable called file_content
                # Hence, this agent can be called easily by other classes
                {"role": "user", "content": f"Analyse the following file content: {file_content}"}
            ]
        )

        # Check if the API returned a valid response, if not, return an error message
        if completion.choices and completion.choices[0].message.content:
            return completion.choices[0].message.content
        else:
            return "Error: The API returned an empty response. Please try again later."

    except Exception as e:
        return f"Error: An exception occurred while analyzing the file: {str(e)}"
```

Libraries:

- OpenAI
- Os
- Dotenv

Key function: analyze_file (OpenAI.com, N.D.)

File Archival Agent

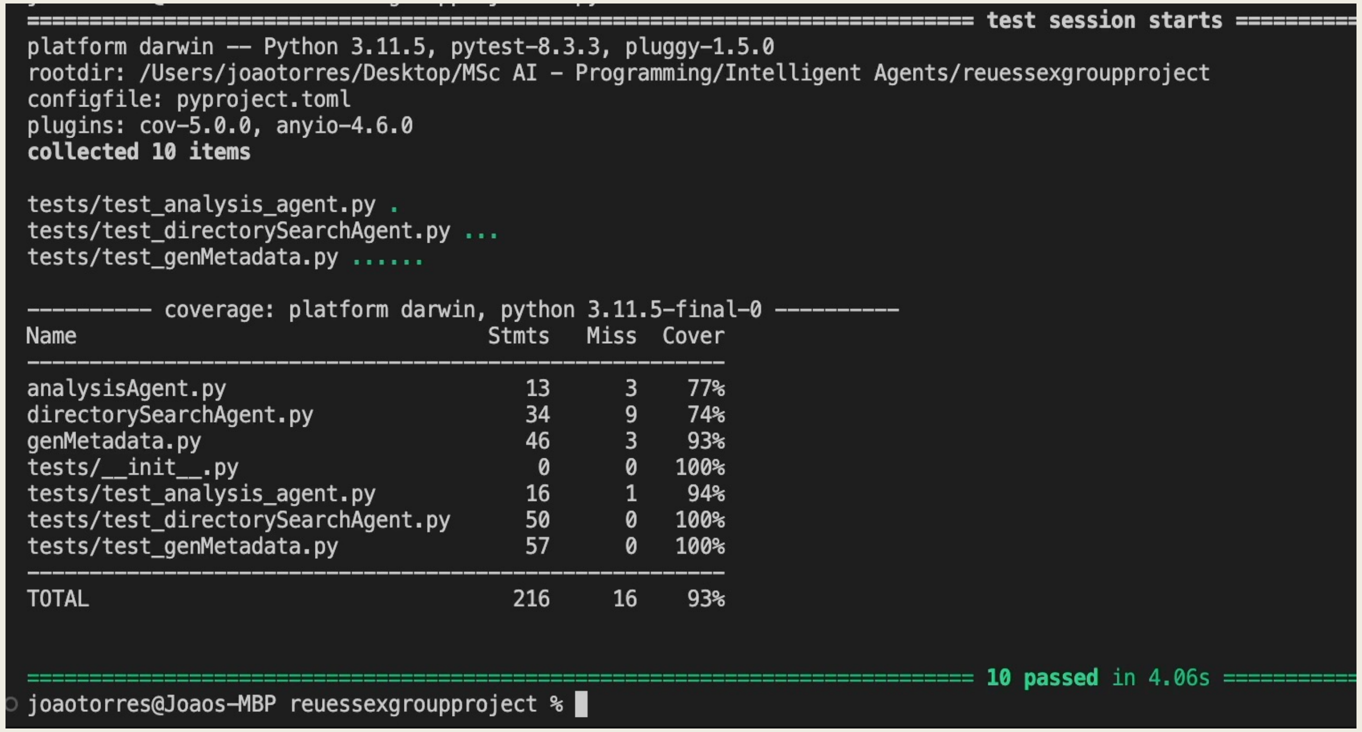


```python
class ArchiveFiles:
    # A class to archive files from a source directory to an archive directory. OOP approach to enhance agent like arquitecture

    def __init__(self, source_dir, archive_dir):
        # Initialize the ArchiveFiles object, where source_dir is a list of source file paths and archive_dir is the destination directory for archiving
        self.source_dir = source_dir
        self.archive_dir = archive_dir

    def archivefiles(self, move=1):
        # Archive files by either moving or copying them to the archive directory.
        # move is a boolean that indicates if the files should be moved or copied.
        # Default is 1 (move) because in a forensic context, we want to move the files to the archive directory to keep the original directory clean
        for file_path in self.source_dir:
            if move == 1:
                shutil.move(file_path, self.archive_dir)
            else:
                shutil.copy(file_path, self.archive_dir)
```

Library: Shutil

Key function: archivefiles

Unit + Integration Testing



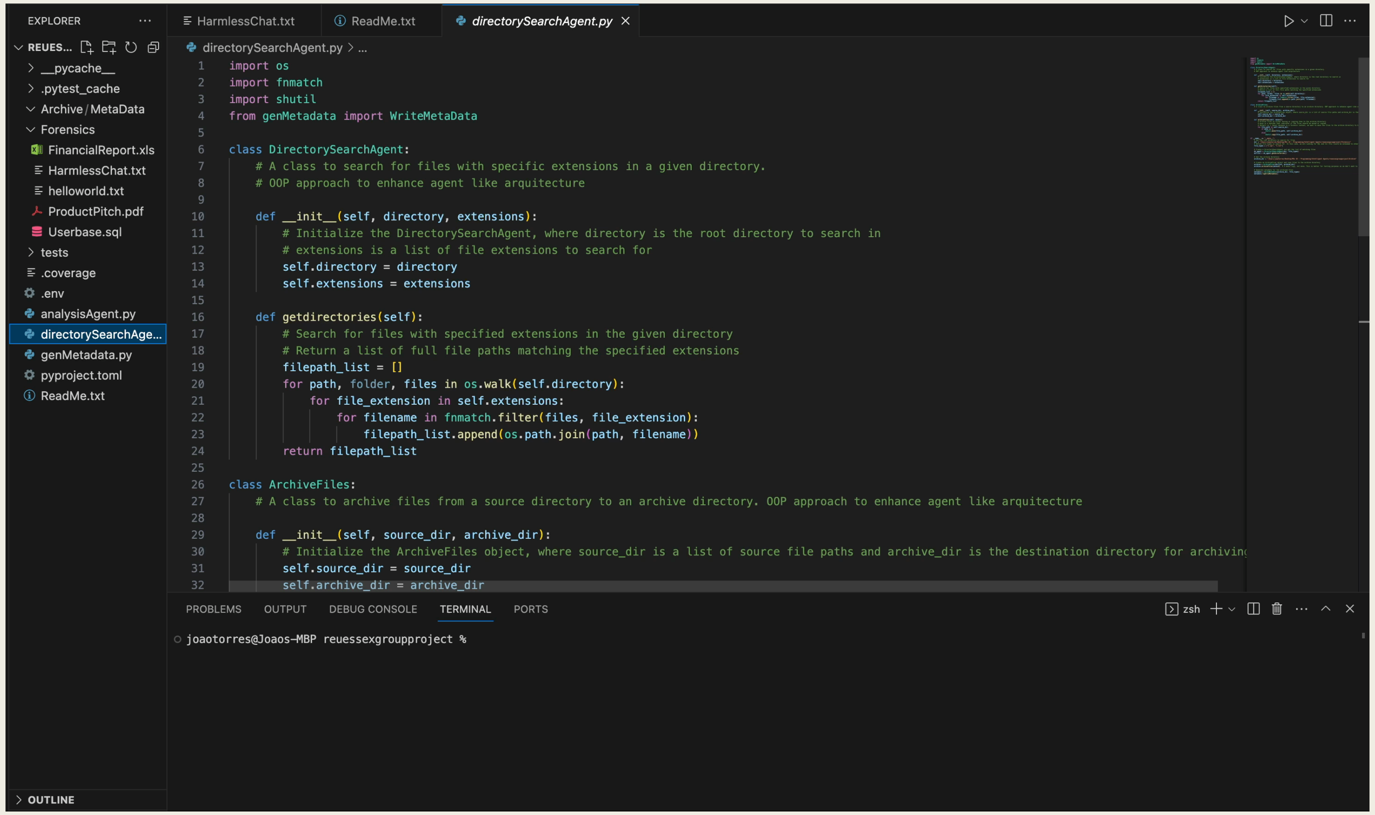
```
============================================================================ test session starts ==========
platform darwin -- Python 3.11.5, pytest-8.3.3, pluggy-1.5.0
rootdir: /Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject
configfile: pyproject.toml
plugins: cov-5.0.0, anyio-4.6.0
collected 10 items

tests/test_analysis_agent.py .
tests/test_directorySearchAgent.py ...
tests/test_genMetadata.py ......

---------- coverage: platform darwin, python 3.11.5-final-0 ----------
Name                                    Stmts   Miss  Cover
-----------------------------------------------------------
analysisAgent.py                           13      3    77%
directorySearchAgent.py                    34      9    74%
genMetadata.py                             46      3    93%
tests/__init__.py                           0      0   100%
tests/test_analysis_agent.py               16      1    94%
tests/test_directorySearchAgent.py         50      0   100%
tests/test_genMetadata.py                  57      0   100%
-----------------------------------------------------------
TOTAL                                     216     16    93%


============================================================================ 10 passed in 4.06s ==========
joaotorres@Joaos-MBP reuessexgroupproject %
```

Library: Pytest (Pytest.org, N.D.)

Results:

- 3 to 4s Runtime
- 10 tests run, 10 passed
- 83% Average coverage on Agent .py files
- 93% Global coverage

Functional Demo (Harmless files)



```python
import os
import fnmatch
import shutil
from genMetadata import WriteMetaData

class DirectorySearchAgent:
    # A class to search for files with specific extensions in a given directory.
    # OOP approach to enhance agent like arquitecture

    def __init__(self, directory, extensions):
        # Initialize the DirectorySearchAgent, where directory is the root directory to search in
        # extensions is a list of file extensions to search for
        self.directory = directory
        self.extensions = extensions

    def getdirectories(self):
        # Search for files with specified extensions in the given directory
        # Return a list of full file paths matching the specified extensions
        filepath_list = []
        for path, folder, files in os.walk(self.directory):
            for file_extension in self.extensions:
                for filename in fnmatch.filter(files, file_extension):
                    filepath_list.append(os.path.join(path, filename))
        return filepath_list

class ArchiveFiles:
    # A class to archive files from a source directory to an archive directory. OOP approach to enhance agent like arquitecture

    def __init__(self, source_dir, archive_dir):
        # Initialize the ArchiveFiles object, where source_dir is a list of source file paths and archive_dir is the destination directory for archivin
        self.source_dir = source_dir
        self.archive_dir = archive_dir
```

```
joaotorres@Joaos-MBP reuessexgroupproject %
```

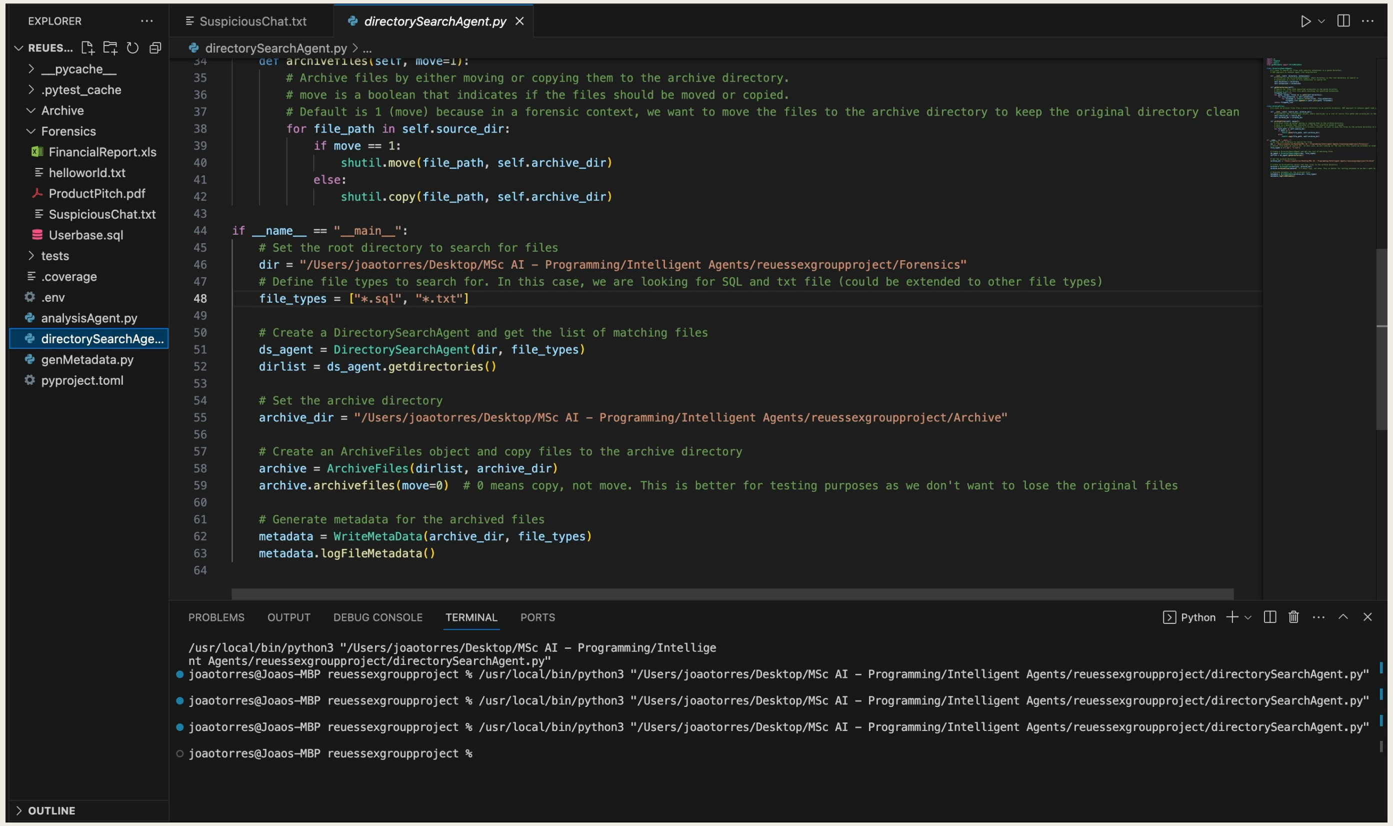
Preconditions:

- Set Forensics and Archive folders
- Define types of files that should be analyzed

Features:

- Searches Matching files in folder
- Gets MetaData
- AI Agent summarizes content of files and defines Criminal Likelihood score

Functional Demo (Suspicious file)



```python
    def archivefiles(self, move=1):
        # Archive files by either moving or copying them to the archive directory.
        # move is a boolean that indicates if the files should be moved or copied.
        # Default is 1 (move) because in a forensic context, we want to move the files to the archive directory to keep the original directory clean
        for file_path in self.source_dir:
            if move == 1:
                shutil.move(file_path, self.archive_dir)
            else:
                shutil.copy(file_path, self.archive_dir)

if __name__ == "__main__":
    # Set the root directory to search for files
    dir = "/Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject/Forensics"
    # Define file types to search for. In this case, we are looking for SQL and txt file (could be extended to other file types)
    file_types = ["*.sql", "*.txt"]

    # Create a DirectorySearchAgent and get the list of matching files
    ds_agent = DirectorySearchAgent(dir, file_types)
    dirlist = ds_agent.getdirectories()

    # Set the archive directory
    archive_dir = "/Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject/Archive"

    # Create an ArchiveFiles object and copy files to the archive directory
    archive = ArchiveFiles(dirlist, archive_dir)
    archive.archivefiles(move=0)  # 0 means copy, not move. This is better for testing purposes as we don't want to lose the original files

    # Generate metadata for the archived files
    metadata = WriteMetaData(archive_dir, file_types)
    metadata.logFileMetadata()
```

```
/usr/local/bin/python3 "/Users/joaotorres/Desktop/MSc AI - Programming/Intellige
nt Agents/reuessexgroupproject/directorySearchAgent.py"
● joaotorres@Joaos-MBP reuessexgroupproject % /usr/local/bin/python3 "/Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject/directorySearchAgent.py"

● joaotorres@Joaos-MBP reuessexgroupproject % /usr/local/bin/python3 "/Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject/directorySearchAgent.py"

● joaotorres@Joaos-MBP reuessexgroupproject % /usr/local/bin/python3 "/Users/joaotorres/Desktop/MSc AI - Programming/Intelligent Agents/reuessexgroupproject/directorySearchAgent.py"

○ joaotorres@Joaos-MBP reuessexgroupproject %
```

Different Result: The AI Agent detects the suspicious conversation included in one of the files.

Reference List

- F. Casino et al. (2022) Research Trends, Challenges, and Emerging Topics in Digital Forensics: A Review of Reviews. *IEEE Access* 10: 25464-25493. DOI: 10.1109/ACCESS.2022.3154059
- OpenAI.com (N.D.) Developer Quickstart. Available from: https://platform.openai.com/docs/quickstart [Accessed: 10 October 2024]
- Pytest.org (N.D.) Pytest Documentation. Available from: https://docs.pytest.org/en/stable/contents.html [Accessed: 11 October 2024]