

# Enhancing Digital Forensics Investigation Process through Large Language Model

K.K.S. Fernando

L. M. F. Hussain

W.A.T.S Jayathilaka

2025



# Enhancing Digital Forensics Investigation Process through Large Language Model

K.K.S. Fernando

Index No: 20020384

L. M. F. Hussain

Index No: 20020473

W.A.T.S Jayathilaka

Index No: 20020511

Supervisor : Dr. Asanka.P. Sayakkara

Mentor : Akila Wickramasekara

May 2025

Submitted in partial fulfillment of the requirements of the B.Sc.  
(Honours) Bachelor of Science in Information Systems Final Year Project



# Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: K.K.S. Fernando



.....  
Signature of Candidate

Date : 26-June-2025

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: L. M. F. Hussain



.....  
Signature of Candidate

Date : 26-June-2025

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: W.A.T.S Jayathilaka

A handwritten signature in black ink, appearing to read 'W.A.T.S Jayathilaka', with a horizontal line underneath.

.....  
Signature of Candidate

Date : 26-June-2025

This is to certify that this dissertation is based on the work of,

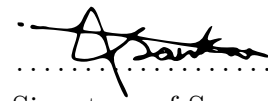
K.K.S. Fernando

L. M. F. Hussain

W.A.T.S Jayathilaka

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Principle/Supervisor's Name: Dr. Asanka. P. Sayakkara

  
.....

Signature of Supervisor

Date : 2025-06-30

# Abstract

Digital forensics, particularly file system analysis, often requires specialized knowledge and complex tools like The Sleuth Kit (TSK), which can hinder the efficiency of investigations. To address this, we propose a framework that leverages Large Language Models (LLMs) and AI agents to interpret natural language instructions and automate TSK commands. By automating complex forensic tasks through natural language processing, our framework has the potential to significantly accelerate investigations, reduce the likelihood of errors, and democratize access to advanced forensic tools. This approach aims to make digital forensics more accessible, efficient, and user-friendly for a broader range of investigators.

The research addresses key challenges in digital forensics, such as the complexity of command-line tools and the need for rapid, accurate analysis in time-sensitive cases. Through a Design Science Research Methodology (DSRM), the framework was developed and refined by establishing a secure environment with local LLMs and AI agents, enhanced by Retrieval-Augmented Generation (RAG) and ReAct prompting to mitigate LLM hallucination and improve reasoning, implementing a user-friendly chat interface with logging and case tracking features based on practitioner feedback, and integrating tool-calling functionality for dynamic access to TSK documentation, further improving accuracy and flexibility.

Quantitative evaluation using AutoDFBench demonstrated that the framework achieved an average precision of 64.74%, recall of 28.70%, and an F1-score of 36.56% across forensic string search tasks. These results indicate that while the framework shows strong precision in executing forensic commands accurately, opportunities remain to further optimize recall and overall retrieval performance. Input from digital forensic experts further confirmed the framework’s effectiveness in streamlining investigations and reducing errors. The multi-agent architecture, with specialized roles for task translation, code writing, and reporting, ensures a modular and efficient approach to forensic analysis.

This work contributes a novel, user-centric solution to digital forensics, making advanced forensic tools more accessible and paving the way for future AI-powered forensic systems.

**Keywords:** digital forensics, file system analysis, AI agents, large language models, The Sleuth Kit, natural language processing, forensic automation, forensic ai assistant

# Acknowledgement

First of all, we would like to express our heartfelt gratitude to our supervisor, Dr. Asanka P. Sayakkara, for his continuous guidance, encouragement, and valuable feedback throughout the course of this research. His insights were instrumental in shaping our work and helping us navigate each stage of the project.

We are especially grateful to Mr. Akila Wickramasekara and Prof. Mark Scanlon from University College Dublin, Ireland, for their mentorship and support throughout this research. Their expert advice, insightful suggestions, and dedicated involvement played a crucial role in refining our approach. We would also like to extend our sincere thanks to Prof. Scanlon for generously providing us with server resources, which greatly supported the technical implementation and testing phases of our framework.

We also acknowledge the academic staff and lecturers at the University of Colombo School of Computing for providing us with the foundational knowledge and guidance that supported this work.

Special thanks go to our families and friends for their endless patience, motivation, and support during the entire research process. Their encouragement helped us stay focused and committed to our goals.

Finally, we appreciate all individuals who contributed directly or indirectly by sharing ideas, offering feedback, or assisting with testing. Your contributions have added great value to the quality of this thesis.



# Contents

<b>Declaration</b>	<b>1</b>
<b>Abstract</b>	<b>4</b>
<b>Acknowledgement</b>	<b>5</b>
<b>List of Figures</b>	<b>9</b>
<b>List of Tables</b>	<b>10</b>
<b>List of Acronyms</b>	<b>11</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	1
1.2 Research Questions . . . . .	1
1.3 Goals and Objectives . . . . .	2
1.3.1 Goals . . . . .	2
1.3.2 Objectives . . . . .	2
1.4 Research Approach . . . . .	2
1.5 Limitations, Scope and Assumptions . . . . .	3
1.5.1 Limitations . . . . .	3
1.5.2 Scope . . . . .	3
1.5.3 Out of Scope . . . . .	3
1.5.4 Assumption . . . . .	4
1.6 Contribution . . . . .	4
<b>2 Background and Literature Review</b>	<b>5</b>
2.1 Background . . . . .	5
2.1.1 Large Language Models (LLMs) . . . . .	5
2.1.2 AI Agents and Multi-Agent Systems . . . . .	5
2.1.3 Natural Language Processing (NLP) . . . . .	5
2.1.4 Prompt Engineering . . . . .	5
2.1.5 Retrieval-Augmented Generation (RAG) . . . . .	6
2.1.6 AutoGen Framework . . . . .	6
2.1.7 Computer File System Forensics . . . . .	6
2.1.8 Research Relevance . . . . .	6

2.1.9	ReAct Prompt Engineering . . . . .	6
2.2	Literature Review . . . . .	7
<b>3</b>	<b>Methodology</b>	<b>11</b>
3.1	First Iteration . . . . .	11
3.1.1	Problem Identification and Motivation . . . . .	11
3.1.2	Objectives . . . . .	12
3.1.3	Design and Development . . . . .	13
3.1.4	Demonstration . . . . .	13
3.1.5	Evaluation via Expert Feedback . . . . .	13
3.1.6	Implementation Setup . . . . .	13
3.1.7	Conclusion of First Iteration . . . . .	14
3.2	Second Iteration . . . . .	14
3.2.1	Problem Identification and Motivation . . . . .	14
3.2.2	Objectives . . . . .	15
3.2.3	Design and Development . . . . .	15
3.2.4	Demonstration . . . . .	16
3.2.5	Evaluation via Expert Feedback . . . . .	16
3.2.6	Implementation Setup . . . . .	16
3.2.7	Conclusion of Second Iteration . . . . .	16
3.3	Third Iteration . . . . .	17
3.3.1	Problem Identification and Motivation . . . . .	17
3.3.2	Objectives . . . . .	18
3.3.3	Design and Development . . . . .	18
3.3.4	Demonstration with NIST Disk Image . . . . .	19
3.3.5	Evaluation via AutoDFBench . . . . .	19
3.3.6	Implementation Setup . . . . .	19
3.3.7	Conclusion of Third Iteration . . . . .	20
3.4	Communication . . . . .	21
<b>4</b>	<b>Results and Evaluation</b>	<b>22</b>
4.1	Qualitative Evaluation . . . . .	22
4.1.1	Introduction . . . . .	22
4.1.2	Methodology . . . . .	22
4.1.3	Findings . . . . .	22
4.1.4	Conclusion . . . . .	23
4.2	Quantitative Evaluation . . . . .	24
4.2.1	Introduction . . . . .	24
4.2.2	Integration of AutoDFBench with the AI Agent Framework . . . . .	24
4.2.3	Methodology . . . . .	25
4.2.4	Results . . . . .	27
4.2.5	Discussion . . . . .	29

4.3	Conclusion of Evaluation . . . . .	30
<b>5</b>	<b>Discussion and Conclusion</b>	<b>31</b>
5.1	Research Findings . . . . .	31
5.2	Discussion . . . . .	33
5.3	Final Conclusion . . . . .	35
5.4	Recommendation . . . . .	37
5.5	Future Work . . . . .	37
5.5.1	Multimodal Forensics Integration. . . . .	37
5.5.2	Support to Other Digital Forensic Branches . . . . .	38
5.5.3	Create a Dataset for Fine-Tuning an LLM . . . . .	38
5.5.4	Framework Testing with Fine-Tuned LLMs . . . . .	38
	<b>References</b>	<b>41</b>
	<b>Appendices</b>	<b>42</b>
<b>A</b>	<b>Screenshots of the developed graphical user interface</b>	<b>43</b>
<b>B</b>	<b>Screenshots of Qualitative Evaluation Findings</b>	<b>46</b>
2.1	Overall Satisfaction . . . . .	46
2.2	UI/UX Evaluation . . . . .	47
2.3	Task Breakdown Effectiveness . . . . .	47
2.4	Accuracy of Analysis . . . . .	48
2.5	Clarity of AI Agent Guidance . . . . .	48
2.6	Code Generation & Execution . . . . .	49
2.7	Adoption and Impact . . . . .	49
2.8	Additional Features . . . . .	50
2.9	General Comments & Suggestions . . . . .	50
<b>C</b>	<b>Results of Evaluation</b>	<b>51</b>
<b>D</b>	<b>Github Repository</b>	<b>54</b>

# List of Figures

2.1	Architecture of the Proposed Framework [1] . . . . .	8
3.1	Architecture of the AI-driven DF Framework in the First Iteration . . . . .	14
3.2	Architecture of the AI-driven DF Framework in the Second Iteration . . . . .	17
3.3	Architecture of the AI-driven DF Framework in the Third Iteration . . . . .	20
4.1	grouped bar chart with precision and recall . . . . .	28
4.2	F1 score chart with test cases . . . . .	29
1.1	Full interface . . . . .	43
1.2	Opening chat interface for a chat requiring a case number . . . . .	44
1.3	Populated chat interface . . . . .	44
1.4	Side bar for tracking chats with a case number with the option to search . .	45
1.5	Button to download logs . . . . .	45
1.6	Generated log file . . . . .	45
2.1	Practitioner feedback on Overall Satisfaction . . . . .	46
2.2	Practitioner feedback on UI/UX Evaluation . . . . .	47
2.3	Practitioner feedback on Task Breakdown Effectiveness . . . . .	47
2.4	Practitioner feedback on Accuracy of Analysis . . . . .	48
2.5	Practitioner feedback on Clarity of AI Agent Guidance . . . . .	48
2.6	Practitioner feedback on Code Generation and Execution . . . . .	49
2.7	Practitioner feedback on Adoption and Impact . . . . .	49
2.8	Practitioner requests for Additional Features . . . . .	50
2.9	Practitioner General Comments and Suggestions . . . . .	50
3.1	FT-SS-02 Test case results using auto df bench . . . . .	51
3.2	FT-SS-03 Test case results using auto df bench . . . . .	51
3.3	FT-SS-04 Test case results . . . . .	52
3.4	FT-07-RTL Test case results using auto df bench . . . . .	52
3.5	FT-SS-09 results . . . . .	52
3.6	FT-SS-10 results . . . . .	53

# List of Tables

4.1	Performance Metrics for Forensic String Search Test Cases . . . . .	27
4.2	Confusion Matrix for Forensic String Search Test Cases . . . . .	28

# List of Acronyms

**AI** Artificial Intelligence

**CDF** Center for Digital Forensics

**CFTT** Computer Forensics Tool Testing Program

**DF** Digital Forensics

**DSRM** Design Science Research Methodology

**LLMs** Large Language Models

**MAS** Multi-Agent Systems

**NIST** National Institute of Standards and Technology

**NLP** Natural Language Processing

**RAG** Retrieval-Augmented Generation

**ReAct** Reason + Act

**TSK** The Sleuth Kit

**UCSC** University of Colombo School of Computing

**UI/UX** User Interface/User Experience

# Chapter 1

## Introduction

### 1.1 Problem Statement

With the rapid advancement of technology, digital devices have become an essential part of everyday life. As a result, digital crimes have significantly increased over the years, presenting serious challenges for law enforcement and forensic investigators. The growing number of digital crime cases has led to the need for more efficient investigation methods that can handle large volumes of data while maintaining accuracy and reliability.

File system forensics, a core area of digital forensics, involves analyzing file systems to recover, identify, and interpret digital evidence stored on storage media. Tools such as The Sleuth Kit (TSK) are widely used in this domain due to their powerful capabilities and flexibility [2]. Although forensic experts are trained to operate such tools, some commands are highly specialized or infrequently used. As a result, investigators often need to consult official documentation or references to ensure proper usage. This process can slow down the investigation and introduce the possibility of delays or mistakes, especially during time-sensitive or large-scale forensic tasks.

These challenges highlight the need for an intelligent, accessible, and automated approach to support forensic experts in conducting investigations more efficiently. A system that can interpret natural language input and handle forensic operations through AI-driven assistant can help streamline the process, reduce manual effort, and support consistent and accurate digital investigations.

### 1.2 Research Questions

To address the problems identified above, this research aims to answer the following questions:

- 1. Does the integration of LLMs into the investigation process reduce the demand for technical expertise of DF practitioners?**

This question explores the specific mechanisms by which LLMs can simplify the use of DF tools to perform forensic tasks effectively.

## 2. How to effectively implement the proposed AI-driven DF framework as a user-friendly tool for real-world forensic investigations?

The goal of this research question is to explore the practical steps and methodologies required to transform the theoretical framework into a functional tool that can be easily used by DF investigators in real-world scenarios.

## 1.3 Goals and Objectives

### 1.3.1 Goals

The primary goal of this research is to design and develop a user-friendly and intelligent framework that assists digital forensic investigators in performing file system forensic tasks using natural language instructions. This is achieved by integrating LLMs and AI agents with TSK to automate the execution of complex commands.

### 1.3.2 Objectives

The specific objectives of the research are as follows:

- To explore the limitations and challenges of existing file system forensic tools and identify areas that can benefit from automation.
- To design and implement AI agents capable of translating natural language queries into TSK commands.
- To develop a user interface that allows investigators to interact with the system in a more intuitive and accessible manner.
- To evaluate the performance of the proposed system in terms of speed, accuracy, and usability through practical scenarios and user feedback.

## 1.4 Research Approach

This research adopts a design science methodology where the solution is developed iteratively and refined through continuous feedback and evaluation. The framework is built using multiple AI agents powered by open-source LLMs, each responsible for different tasks such as task translation, command execution, and report generation. The agents collaborate to process investigator input, translate it into TSK commands, run the commands, and present the output in a structured format. Natural Language Processing (NLP) capabilities are integrated to allow users to issue queries using simple, natural language, reducing the need to memorize syntax. The evaluation strategy includes standard forensic tasks such as file recovery and metadata extraction, using precision, recall, F1-score, and usability metrics. Feedback from digital forensic practitioners is also used to measure the practical effectiveness and usability of the system [1, 3].



## 1.5 Limitations, Scope and Assumptions

### 1.5.1 Limitations

This research has several limitations and ethical considerations that should be acknowledged. The framework relies on Large Language Models (LLMs), which can sometimes generate incorrect or misleading outputs, especially when exposed to lengthy or ambiguous prompts. Although methods such as Retrieval-Augmented Generation (RAG) and ReAct prompting were used to reduce these risks, final outputs still require careful human review.

Another limitation concerns the secure use of LLMs in forensic environments. To maintain the forensic soundness of evidence and ensure data privacy, the LLMs were deployed locally using isolated environments. This setup prevents sensitive data from being transmitted across external networks and confines command execution through predefined functions to avoid potential misuse.

The datasets used during development and testing were publicly available and standardized, such as those provided by NIST CFTT. Handling real-world case data would require additional ethical and legal considerations, including data protection, chain of custody, and institutional clearance, which were beyond the scope of this study.

### 1.5.2 Scope

#### 1.5.2.1 In scope

1. **AI Agents for File System Forensics::** Development of specialized agents, including a Chat Manager, Task Translation Assistant, Coder Agent, and Reporter Agent, specifically for file system forensics tasks. Each agent will perform unique roles to streamline TSK command execution and data handling.
2. **User interface:** Developing a UI to input natural language queries and allow the user to step in and change the agents' actions when needed.

#### 1.5.3 Out of Scope

1. **Fine tuned AI Agent:** The project will not involve agents specially trained for advanced analysis tasks outside the general scope of file system forensics.
2. **Cross-Domain Agent Training:** The framework is limited to file system forensics and does not extend to other DF branches, such as network or memory forensics.
3. **Advanced Customization of TSK:** Existing TSK commands are utilized without modifying or creating custom modules specific to this project.
4. **Evaluation of Agent Frameworks:** This research does not aim to evaluate or compare the performance of the AutoGen framework against other AI agent orchestration frameworks.

5. **Anti-Forensics Considerations:** This study does not address anti-forensics techniques or scenarios where evidence may be intentionally altered or concealed.

#### 1.5.4 Assumption

It is assumed that users of the proposed system possess a basic understanding of digital forensic principles and are familiar with general practices in forensic investigations. While the system is designed to simplify the use of forensic tools, it still requires users to have a foundational awareness of how file system forensics works and how command-line tools like TSK are typically used. Users are expected to understand the context of forensic tasks, such as file recovery or metadata analysis, in order to interpret and validate the outputs effectively.

Additionally, it is assumed that the built-in functionalities of TSK are adequate for performing the types of forensic operations required within the scope of this research. The system leverages these existing features without extending or modifying TSK itself. It is also assumed that the selected open-source LLMs are capable of performing reliable command translation and task automation using natural language instructions, without the need for model fine-tuning.

### 1.6 Contribution

This research makes several key contributions to the digital forensics field. Firstly, it presents a novel AI-powered framework that significantly reduces the complexity involved in using forensic tools like TSK by allowing natural language interaction. Secondly, it introduces a multi-agent architecture where each AI agent performs a dedicated role, enabling a modular and efficient approach to forensic investigations [3]. Thirdly, it provides a comprehensive evaluation of the proposed system, measuring its impact on investigation speed, accuracy, and ease of use. Lastly, the research addresses the gap in accessibility and usability of forensic tools, especially in environments where technical expertise is limited. By doing so, it lays the groundwork for future advancements in intelligent, user-friendly digital forensic systems that can be adapted for broader forensic tasks beyond file system analysis.

# Chapter 2

## Background and Literature Review

### 2.1 Background

#### 2.1.1 Large Language Models (LLMs)

Large Language Models (LLMs) are a type of artificial intelligence trained on large text datasets to understand and generate human-like language. These models can perform a wide range of natural language processing tasks, such as text generation, summarization, and question answering. Their capability to understand context and provide meaningful responses makes them suitable for tasks that traditionally required human intervention.

#### 2.1.2 AI Agents and Multi-Agent Systems

Autonomous agents are independent systems capable of sensing and interacting with their environments and use the information they gain to independently make their own rational decisions [4]. When organized as Multi-Agent Systems (MAS), these agents can specialize in different tasks and collaborate to complete complex operations. MAS systems are known for improving adaptability, task distribution, and contextual understanding [5, 6].

#### 2.1.3 Natural Language Processing (NLP)

NLP is a field of AI that focuses on the interaction between computers and human language. It allows machines to understand, interpret, and generate human language in a way that is both meaningful and useful. In this research, NLP is used to interpret user input in natural language and translate it into forensic commands understood by the system.

#### 2.1.4 Prompt Engineering

Prompt engineering refers to the practice of designing and refining input queries to guide the behavior of LLMs effectively. Since LLMs are sensitive to the structure and clarity of prompts, this technique is important in ensuring that responses are accurate, relevant, and aligned with user intentions. Strategies like zero-shot and few-shot prompting are often used to improve output consistency and reliability [7].

### **2.1.5 Retrieval-Augmented Generation (RAG)**

RAG is a hybrid approach that combines document retrieval with text generation. It enables the model to retrieve relevant context from a knowledge base or dataset before generating a response, leading to more informed and accurate outputs. In digital forensics, RAG helps improve the reliability of LLM-generated content by anchoring it to verified information[8].

### **2.1.6 AutoGen Framework**

AutoGen is an open-source framework that supports the development of multi-agent LLM applications. It allows multiple agents with different roles and capabilities to work collaboratively on complex tasks while supporting human-in-the-loop control <sup>3</sup>. This framework is highly relevant to this research as it supports agent orchestration and customizable workflows suitable for digital forensic applications.

### **2.1.7 Computer File System Forensics**

File system forensics is the process of examining and analyzing data stored within a computer’s file system to uncover, document, and interpret digital evidence. It involves detailed inspection of file structures, metadata, and storage volumes to recover hidden or deleted files. TSK (The Sleuth Kit) is a widely used open-source tool in this domain. It allows analysts to examine disk images across multiple file system formats and provides critical support for evidence analysis <sup>2</sup>. This research focuses on automating TSK’s command-line interface to improve efficiency and usability.

### **2.1.8 Research Relevance**

By combining these advanced technologies, LLMs, AI agents, AutoGen, and file system forensic tools—this research aims to bridge the gap between complex forensic tools and the growing need for accessibility and efficiency in investigations. The proposed system provides a new approach to forensic analysis by enabling investigators to use natural language to conduct advanced forensic operations with minimal technical barriers.

### **2.1.9 ReAct Prompt Engineering**

ReAct (Reason + Act) is a prompt-based approach introduced to combine reasoning and acting in large language models (LLMs). Unlike traditional prompting methods that focus either on generating reasoning traces (as in Chain-of-Thought prompting) or on action execution, ReAct allows LLMs to interleave reasoning steps with concrete actions. This interleaving enables models to dynamically formulate plans, update them based on new information, and interact with external tools or environments during task execution [9].

In this approach, models generate “thoughts” (reasoning traces) that help guide decisions and determine the next appropriate action. These actions may involve querying APIs, navigating environments, or interacting with data sources. The synergy between internal

reasoning and external action allows ReAct to be more grounded and effective, reducing hallucinations and improving performance across tasks such as question answering, decision making, and information retrieval.

In the context of this research, ReAct prompting is used to enhance the reliability and task execution capabilities of LLM agents when performing forensic operations, such as chaining together forensic tool commands or adapting responses based on intermediate results.

## 2.2 Literature Review

Recent literature has demonstrated significant advancements in applying artificial intelligence (AI) and automation within digital forensics (DF). As digital crime volumes increase and evidence becomes more complex and distributed, the demand for intelligent, adaptive solutions has grown. A growing body of research is exploring how Large Language Models (LLMs) and AI agents can be employed to reduce the technical burden on investigators and automate routine forensic tasks.

Wickramasekara and Scanlon [1] proposed a comprehensive multi-agent framework that integrates LLMs and AutoGen to support key stages of the digital forensic process—specifically the Examination, Analysis, and Reporting phases. Their model defines specific roles for agents, such as task translators, code generators, and reporters. These agents work in coordination, enabling the overall system to interact more efficiently with traditional digital forensic tools, demonstrating how role-based multi-agent design can enhance investigation workflows.

The architecture of this framework (see Figure 2.1) includes a human agent who interacts with a Chat Manager through natural language. The Chat Manager assigns tasks either to a Task Translation Assistant (TTA) or a Reporter Agent based on the type of input. The TTA, powered by a fine-tuned LLM, breaks down complex instructions into subtasks and passes them to the Coder Agent, which is responsible for generating Python code and running tests. Once the output is produced, the Reporter Agent compiles a readable report and sends it back to the Chat Manager, who communicates the results to the human agent. This setup enables dynamic role assignment and effective task orchestration among AI agents, improving usability and reducing manual overhead in forensic workflows.

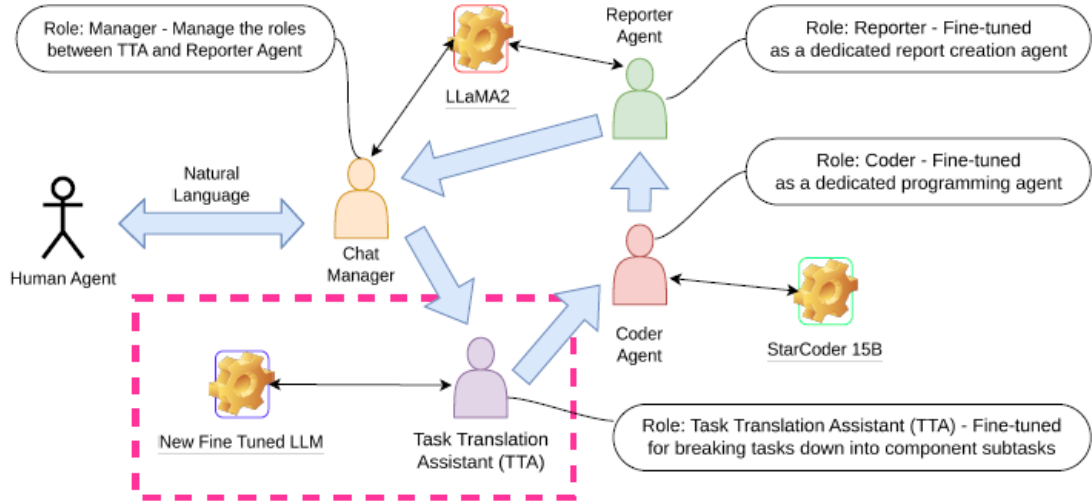


Figure 2.1: Architecture of the Proposed Framework [1]

In earlier efforts toward forensic tool automation, Garfinkel [10] introduced a solution that processes disk images using The Sleuth Kit (TSK) and generates detailed XML output, including metadata such as file names, timestamps, and hash values. This system simplifies the use of TSK by abstracting its command-line complexity and supports integration with scripting tools like Python for large-scale or customized forensic processing. The developed system is then used to create programs for case studies that can be done using TSK. The case studies include Remote File Discovery and Extraction, an Image Redaction Tool and a USB Transfer Kiosk. The paper then provides qualitative comparisons highlighting efficiency and ease of use compared to running regular TSK commands.

In the context of AI, Kakalis and Kefalidis [11] explored the use of open-source LLMs to convert natural language input into SPARQL queries for knowledge graph access. The study focused specifically on small scale open source LLM's and follows a systematic approach to select the model that meets the hardware limitation. The study goes through three type of prompt engineering techniques evaluating them though accuracy.

In a broader AI context, Hymel et al. [6] investigated multi-agent configurations in commercial software environments. Their findings highlight that enabling context-sharing between agents, such as a requirements agent and a code-writing agent, leads to improved relevance and performance. The study evaluates the effect of context sharing through the accepted suggestions from the code-writer agent with 101 developers participating in the experiment. These insights lend support to similar collaborative agent structures being explored in digital forensic applications.

While these studies showcase the potential of agent-based design, command abstraction, and natural language interaction, they each focus on individual aspects of the broader problem. There remains a notable gap in research that unifies these ideas into a complete, automated forensic framework specifically targeting file system forensics. This thesis aims to address that gap by developing an AI-powered system that automates TSK operations through a natural language interface and multi-agent coordination, making forensic tasks

more accessible, efficient, and scalable.

To complement these advancements in forensic automation, Wickramasekara et al. also introduced AutoDFBench, a benchmarking framework designed to test and validate digital forensic tools and scripts generated by large language models (LLMs) [12]. The framework follows a structured testing pipeline involving data preparation, code generation, execution, and result validation based on ground truth datasets from the NIST Computer Forensics Tool Testing Program (CFTT). It calculates evaluation metrics such as precision, recall, and F1 scores to produce a comprehensive AutoDFBench score. The benchmark allows both iterative development and comparative analysis of AI-assisted forensic tools, addressing the current gap in standardised evaluation for generative forensic applications. In this research, AutoDFBench is used to evaluate the effectiveness and reliability of the proposed LLM-powered forensic framework by testing its performance on string search tasks using standard forensic test images.

Together, these works provide a strong foundation for the development and evaluation of the LLM-driven, agent-based forensic framework proposed in this thesis. While these frameworks are promising, recent literature also investigates the limitations and risks of adopting LLMs in forensic environments.

One of the most comprehensive studies is by Scanlon et al., who assessed ChatGPT (GPT-4) across a range of digital forensic tasks—including artifact identification, anomaly detection, keyword search generation, and incident response [13]. Their findings highlighted both strengths and limitations. On the one hand, ChatGPT was able to assist with generating useful code snippets and aiding preliminary investigation steps. On the other, the authors raised critical concerns about hallucinations, outdated knowledge, legal risks, and a lack of domain-specific training. They argued that non-deterministic and non-reproducible outputs make such models unsuitable for high-stakes forensic investigations requiring strict validation.

A. Wickramasekara et al. expanded this perspective in their Systematization of Knowledge (SoK) paper [14]. Their review identified a wide spectrum of use cases for LLMs across the forensic investigation lifecycle—including anomaly detection, log parsing, and report generation—but concluded that their current effectiveness is limited to subtasks or supportive roles rather than complete workflows. They emphasized the ongoing challenges of explainability, black-box behavior, and the need for structured evaluation protocols, especially for courtroom use.

A recurring concern in the literature is the issue of hallucinations. Because LLMs generate coherent text by predicting language patterns rather than grounding their output in factual data, they are susceptible to fabricating plausible but false statements. In digital forensic applications—where accuracy and verifiability are essential—this risk cannot be overlooked. Both Scanlon et al. and Alghaffi et al. [14, 13] caution that without robust human oversight and formal evaluation tools, LLM-generated content poses legal and operational risks.

Despite these risks, the potential of LLM-driven frameworks remains strong.

Wickramasekara et al.’s AutoGen-based architecture is one of the first to offer a complete, agent-driven pipeline that combines natural language interaction, command decomposition, and structured reporting. Their inclusion of memory mechanisms, role-based agents, and integration with evaluation tools like AutoDFBench showcases how LLM-powered systems can evolve into scalable and robust solutions for file system forensics.

In parallel, usability-focused research by Hargreaves et al. revealed that many practitioners underutilized existing forensic tools—especially those that include AI functionality—due to poor usability, lack of trust in results, and overwhelming training requirements [15]. Respondents cited issues such as difficulty filtering results, poor output clarity, and limited control over how AI-enhanced features behave. The DFPulse 2024 report also highlighted that many practitioners do not frequently engage with academic publications or tools due to internal policy constraints, perceived impracticality, or time limitations. These findings underscore the need for intuitive, explainable, and user-aligned interfaces—precisely what this thesis attempts to contribute through natural language prompting and multi-agent coordination.

In conclusion, the reviewed literature reflects a significant shift toward semi-automated, human-guided AI systems in digital forensics. Rather than aiming for fully autonomous solutions, the emphasis is on developing AI assistants that support and extend the capabilities of human investigators. While isolated tools and techniques offer useful improvements, the future lies in integrated frameworks that balance LLM automation with traceability, expert oversight, and standard-compliant evaluation. The proposed system in this thesis addresses this convergence by leveraging natural language interfaces, modular agent architecture, and AutoDFBench testing to create a usable, transparent, and effective solution for real-world digital forensic investigations.



# Chapter 3

## Methodology

This research adopts the Design Science Research Methodology (DSRM), as introduced by [16], to guide the development and evaluation of an AI-driven digital forensics framework. The framework leverages a multi-agent system to automate forensic tasks using The Sleuth Kit (TSK). Key agents include the Chat Manager, which handles user queries and delegates tasks; the Task Translation Agent, which breaks down queries into subtasks; the Code Writer Agent, which generates TSK commands; the Code Executor Agent, which runs these commands; and the Reporter Agent, which compiles results into reports. These agents interact sequentially: the Chat Manager receives natural language input, passes it to the Task Translation Agent, which coordinates with the Code Writer and Executor Agents, while the Reporter Agent delivers the final output, all supported by RAG and ReAct prompting for improved reasoning. DSRM introduces a structured process for conducting research through the creation and evaluation of IT artifacts that aim to solve identified practical problems, ensuring both practical relevance and advancement of scientific knowledge.

The DSRM process model, as defined by Peffers et al. (2007), involves six key activities: problem identification and motivation, definition of objectives for a solution, design and development, demonstration, evaluation, and communication. Accordingly, this research follows these activities through an iterative process, refining the artifact over three iterations. The refinements to the agents are done within each iteration by changing the system prompts of the agents to get a satisfactory output for a defined user query.

### 3.1 First Iteration

#### 3.1.1 Problem Identification and Motivation

The initial iteration aimed to assess the feasibility of implementing a simplified version of the AI-driven digital forensics framework proposed by Wickramasekara, A. and Scanlon, M. [1]. This prototype excluded components such as a fine-tuned large language model (LLM) and specialized agent skills. The goal was to validate whether a baseline system—centered around file system forensic tasks using The Sleuth Kit (TSK)—could function effectively in a real-world setting.

By demonstrating feasibility at this foundational level, the research sought to uncover practical challenges and inform design decisions for subsequent iterations. This approach

also served to assess the potential of integrating AI agents with local LLMs in a secure and controlled environment.

### 3.1.2 Objectives

#### 3.1.2.1 Environment Setup

The first objective was to establish a secure, privacy-preserving environment to deploy the AI framework. Local instances of open-source LLMs were used to ensure the framework could operate in a closed network, similar to a digital forensic (DF) lab.

To safeguard the host environment, LLM-generated code needed to be executed in isolation to mitigate risks posed by potentially unsafe outputs.

**Process:**

- Select and deploy an open-source LLM locally, considering computational constraints.
- Configure a secure environment on a local machine with sufficient GPU resources.
- Establish containerized execution to run LLM-generated commands in isolation.

#### 3.1.2.2 Framework Implementation

A simplified version of the AI agent-based framework was developed, following the principles set out in Wickramasekara, A. and Scanlon, M.'s [1] design. Key enhancements included:

- **Transparency in reasoning:** Similar to including the research process of a the forensic practitioner, the research process of the LLM to get to conclusions is also needed to be included in the forensics report. For this, Agents adopted the ReAct (Reasoning and Action) prompting technique. This technique make the LLM's output a structured response that forces it to show it's thought process.
- **Knowledge enhancement:** LLM's used in this iteration were small scale LLM's. These LLM's lacked the knowledge of TSK. Hence, the information regarding TSK commands had to be augmented along with the user prompt to generate accurate responses. A Retrieval-Augmented Generation (RAG) component was introduced to address this. This component would take in the user query and augment the information of the most relevant TSK commands. This enriches the user's query to generate accurate responses.

**Process:**

- Implement a multi-agent system using Microsoft Autogen.
- Integrate ReAct prompting for reasoning transparency.
- Develop a RAG to supply the LLM with documentation-based TSK knowledge.

### 3.1.3 Design and Development

The initial artifact combined LLM agents, ReAct-enhanced reasoning, RAG-driven knowledge retrieval, and TSK-based forensic utilities in a secure environment.

#### Key Features:

- Develop the backend using Microsoft Autogen to connect the LLM agents through a group chat conversational pattern.
- Process natural language inputs using the Task Translation Agent to generate subtasks, leveraging ReAct for transparent reasoning.
- Generate TSK code scripts with the Code Writing Agent.
- Retrieve TSK-specific knowledge via the RAG Proxy Agent to ensure accuracy.
- Interpret TSK outputs and present them in a clear, user-friendly format.
- Limit scope to file system forensics tasks, specifically focusing on deleted file recovery using TSK.

### 3.1.4 Demonstration

The framework’s feasibility was tested using a NIST disk image to verify the outputs for a basic file system forensics task, such as identifying allocated partitions and their file system types. Subtask reasoning was observed, and both the final outputs and reasoning steps were documented.

### 3.1.5 Evaluation via Expert Feedback

Demonstration results were shared with the project supervisor, and feedback was collected on:

- Usability of the prototype.
- Clarity of ReAct’s reasoning in the Task Translation and Code Writing Agents.
- Effectiveness of the RAG Proxy Agent’s knowledge retrieval.

Outputs were also compared against the expected results from the NIST image. Feedback was analyzed to identify strengths and areas for improvement in subsequent iterations.

### 3.1.6 Implementation Setup

During the first iteration, development and testing were conducted on the researchers’ personal laptops, each equipped with an Intel Core i7 10th Gen processor, 16 GB of RAM, and an NVIDIA GTX 1660 GPU with 6 GB of VRAM. Due to these limited computational



with the framework, and to gather actionable feedback for improving it further. Based on availability of DF practitioners in Sri Lanka who could have a hands-on with the tool, we were limited to two individuals from the Centre for Digital Forensics (CDF) in UCSC. As such, this was the sample for our evaluations in this iteration. As far as representing the digital forensics practitioner community, this sample does not do the community justice. Even though we got feedback regarding this AI assisted tool, the authors of this paper fully agree with this. What is needed to be considered is that this was the only available resource within our reach due to the time constraints of this research.

### 3.2.2 Objectives

#### 3.2.2.1 Implement a User-Friendly Interface for Feedback Collection

**Process:**

- Develop a basic chat interface.
- Demonstrate how the framework accomplishes a basic DF task using a NIST disk image.
- Obtain feedback from practitioners to enhance the interface and align it with the expectations of a professional DF tool.

### 3.2.3 Design and Development

The development began with an exploration of projects implementing user interfaces with Autogen as the backend. A chat interface was initially developed using Streamlit [17], connected with the backend by overriding the `Assistant` and `UserProxyAgent` classes provided by Autogen. However, limitations in Autogen 0.2—specifically its incompatibility with event-driven and asynchronous UI libraries—led to issues such as deletion of previous chat messages on new human inputs.

Rather than investing excessive effort in mitigating these limitations, two alternative solutions were explored:

- Migrating to Autogen 0.4, which introduces native support for event-driven and asynchronous interactions [18].
- Redeveloping the UI using Chainlit [19], a more robust interface layer for conversational AI applications.

Although the migration to Autogen 0.4 introduced dependency conflicts and major backend changes, the Chainlit-based solution proved feasible. A helper method was used to handle page refresh issues, resulting in a functional and responsive interface.

### 3.2.4 Demonstration

The updated UI was demonstrated to digital forensic practitioners. They emphasized the importance of generating and storing logs as a critical feature of any DF tool—something missing in the initial UI. Through discussion, parallels were drawn between traditional DF processes (e.g., testing hypotheses using multiple tools) and the framework’s ability to run various commands and follow multiple investigative paths.

Practitioners showed particular interest in a feature allowing them to select specific tasks that supported a hypothesis and include them in a log. Additionally, they noted the possibility of extending the framework for proprietary DF tools by updating the RAG documentation. Another significant requirement was the ability to track investigations using a case number, a standard feature in existing DF tools.

### 3.2.5 Evaluation via Expert Feedback

Assess the artifact’s effectiveness and usability based on expert feedback.

**Process:**

- Collect feedback from DF practitioners regarding usability and adherence to standards.
- Share demonstration outcomes with a supervisor.
- Analyze insights to identify strengths and areas for further refinement.

### 3.2.6 Implementation Setup

For the second iteration, the team utilized a desktop computer from the UCSC research lab, featuring an Intel Core i5 8th Gen processor, 32 GB RAM, and an NVIDIA GTX 1080 Ti GPU (11 GB VRAM). The system ran a centralized Ollama server instance hosting the LLM, which was accessed via local Wi-Fi network from developers’ devices by exposing the server port.

This setup allowed for more powerful LLMs, including Qwen 32B, to be loaded and tested. The Ollama server was integrated with a custom user interface built using Chainlit, enabling a more interactive and realistic simulation of how the framework would operate in real-world forensic environments. Although the response time for larger models remained slow, this environment significantly improved development workflow and enabled shared access for multi-user testing.

### 3.2.7 Conclusion of Second Iteration

This iteration successfully introduced a functional user interface. Based on practitioner feedback, two core features were implemented:

- A downloadable log of all chats and agent-generated commands.

- A case tracking feature, allowing users to initiate sessions with a case number and track interaction history accordingly.

These enhancements demonstrate the potential for transforming the prototype into a practical, practitioner-friendly digital forensic tool.

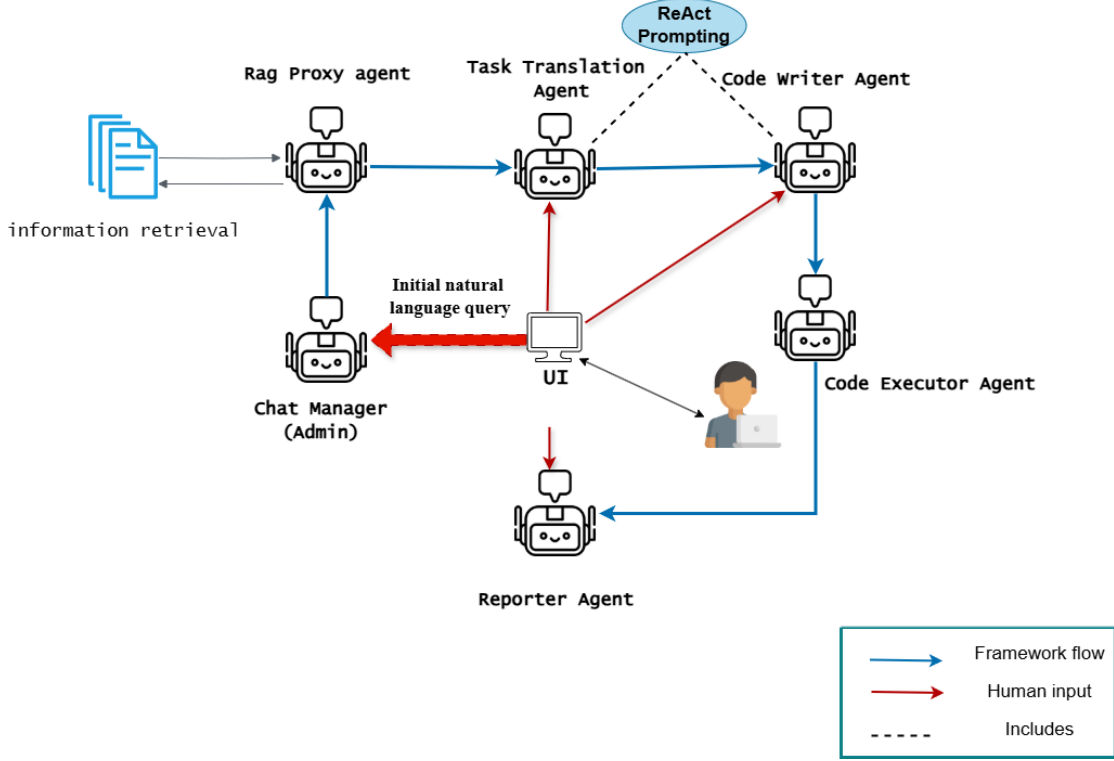


Figure 3.2: Architecture of the AI-driven DF Framework in the Second Iteration

### 3.3 Third Iteration

#### 3.3.1 Problem Identification and Motivation

The third iteration investigates whether the AI-driven DF framework, now enhanced with *tool calling functionality*, can serve as a robust, user-friendly, and valid digital forensics tool for practitioners to efficiently perform file system forensic tasks. The scope remains focused on file system forensics, utilizing The Sleuth Kit (TSK)—a widely recognized open-source forensic toolkit [2].

Building upon the backend framework developed in the first iteration (with ReAct and RAG Proxy Agent) and the user interface introduced in the second iteration (with log generation and case tracking), this iteration focuses on integrating tool calling to dynamically access TSK resources and evaluates the framework’s performance using AutoDFBench.

### 3.3.2 Objectives

#### 3.3.2.1 Integrate Tool Calling Functionality

Enhance the AI agent framework with tool calling functionality to dynamically retrieve TSK documentation and execute forensic tasks, thereby improving task accuracy and agent flexibility.

**Process:**

- Identify forensic tools and documentations relevant to common file system forensic tasks.
- Develop a tool calling mechanism within the agent framework, enabling agents to invoke tools as needed based on task context.
- Implement tools such as `get_tool_documentation`, which can retrieve and parse relevant TSK documentation from web sources.
- Test the integration by ensuring the Task Translation, Code Writing, and RAG Proxy Agents can leverage tool outputs effectively in task execution.

### 3.3.3 Design and Development

Extend the backend framework and user interface to support robust tool invocation for forensic task execution.

**Process:**

- Extend the Python-based backend to support external tool invocation by the agents.
- Develop a suite of domain-specific tools, including:
  - `get_tool_documentation`: Scrapes and parses web-based TSK documentation to extract command functionalities.
  - Additional tools for tasks like metadata extraction and file system analysis.
- Integrate tool calling with the Chainlit-based user interface from the second iteration, enabling natural language triggering of tools.
- Address development challenges such as tool output compatibility by introducing helper functions for standardizing tool responses.
- Conduct functional testing to confirm correct tool invocation and relevance of results to forensic workflows.



### 3.3.4 Demonstration with NIST Disk Image

Demonstrate the framework’s enhanced forensic capabilities using a standardized case.

**Process:**

- Use a disk image from the NIST CFReDS repository [20].
- Perform a file system forensic task via the user interface, leveraging:
  - Task Translation Agent for interpreting human inputs.
  - Code Writing Agent for formulating commands.
  - Tool calling for dynamically retrieving TSK documentation and executing commands.
- Document tool usage, user interface interactions, and command outputs.
- Assess improvements in accuracy and efficiency compared to previous iterations.

### 3.3.5 Evaluation via AutoDFBench

Evaluate the framework’s usability and effectiveness using AutoDFBench—an automated benchmarking tool.

**Process:**

- Configure AutoDFBench with the NIST disk image to test the framework on standardized forensic tasks.
- Evaluate metrics such as:
  - Task accuracy
- Analyze quantitative metrics (e.g. Precision, Recall and F1 scores)

### 3.3.6 Implementation Setup

In the third iteration, the framework was deployed on a high-performance server provided by Prof. Mark Scanlon (University College Dublin), remotely accessed via SSH. The server was equipped with an NVIDIA RTX 4090 GPU (24 GB VRAM), Intel Xeon CPU, and 128 GB RAM. This robust configuration enabled deployment of larger models like LLaMA 3 70B and Qwen 72B using Ollama with GPU acceleration.

The server environment used Docker containers to host the LLM services and tool-calling mechanisms. All components, including Autogen agents, Chainlit UI, and forensic disk image mounts, were containerized and managed through Docker Compose. The tool-calling feature was enhanced to dynamically fetch Sleuth Kit documentation via web scraping, improving the accuracy of agent outputs.

This infrastructure allowed real-time execution of forensic tasks, rapid testing using NIST disk images, and integration with AutoDFBench for final evaluations. The powerful GPU and memory ensured that response times and model accuracy were optimal, greatly advancing the reliability and performance of the framework.

### 3.3.7 Conclusion of Third Iteration

This iteration successfully integrated tool calling functionality into the AI-driven DF framework, particularly the `get_tool_documentation` tool. The system, now combining a robust backend (with ReAct and RAG Proxy Agent), a user-friendly Chainlit interface (with log and case tracking), and dynamic tool invocation, was rigorously tested using AutoDFBench.

Results indicate that the integration of tool calling significantly improved task flexibility, output accuracy, and alignment with real-world digital forensic needs. With this final iteration, the DSR process concludes, producing a validated and practically applicable digital forensics tool.

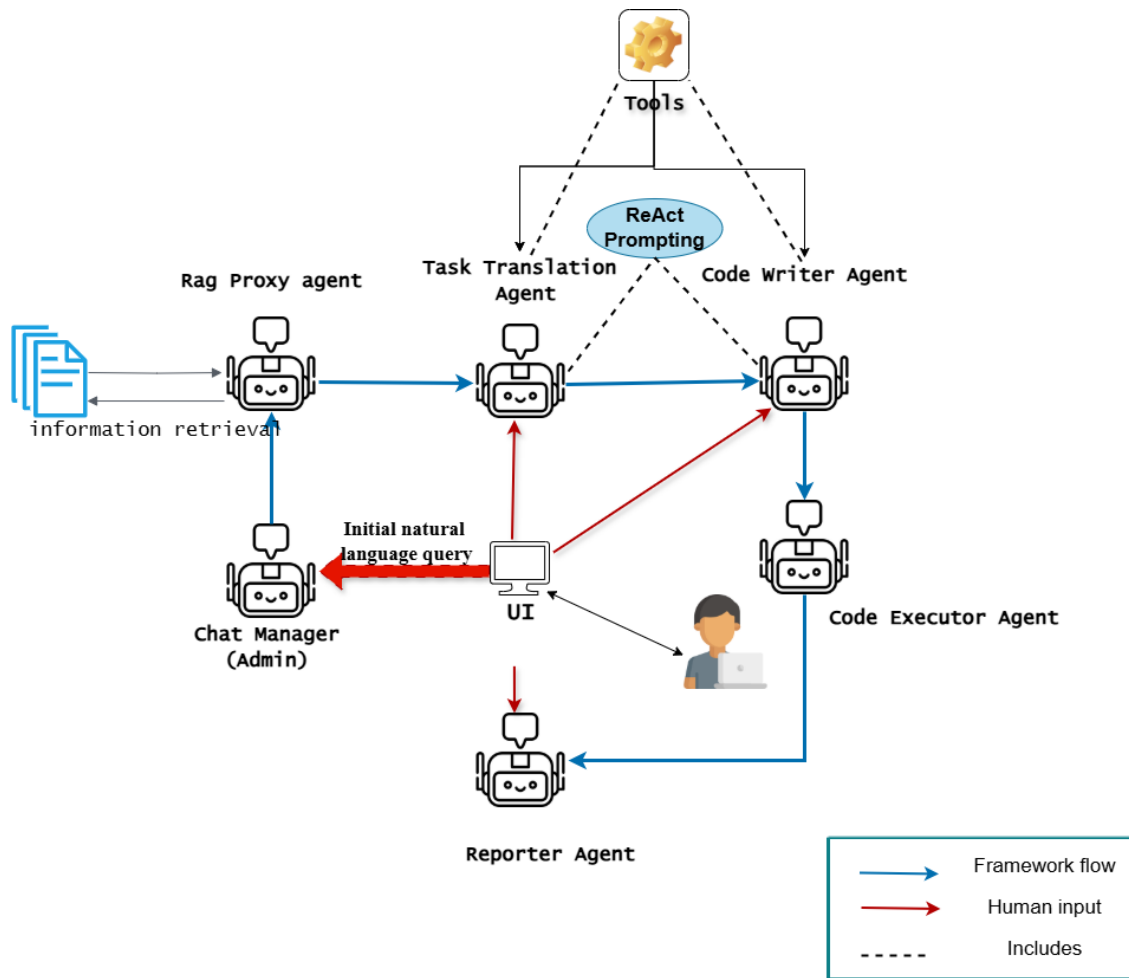


Figure 3.3: Architecture of the AI-driven DF Framework in the Third Iteration

### 3.4 Communication

The final step involves communicating the problem, artifact, its utility, novelty, design, and effectiveness to relevant audiences, including researchers and practitioners. This thesis serves as a primary communication channel for the research findings. The structure follows the DSRM process, detailing each stage from problem definition to evaluation. The research contributes a novel AI-driven framework artifact, demonstrated and evaluated through iterative development and testing, including practitioner feedback and automated benchmarking. The findings on the framework’s potential as a valid and user-friendly tool, particularly the benefits of local LLMs, agent-based architecture with ReAct/RAG, and dynamic tool calling for DF tasks, are presented.

# Chapter 4

## Results and Evaluation

This chapter presents the evaluation of the AI-driven digital forensics framework developed in this research. The evaluation is divided into two parts: a qualitative assessment based on practitioner feedback and a quantitative analysis using the AutoDFBench benchmarking tool. Together, these evaluations assess the framework’s usability, effectiveness, and performance in streamlining digital forensic investigations, particularly for file system analysis using The Sleuth Kit (TSK).

### 4.1 Qualitative Evaluation

#### 4.1.1 Introduction

This qualitative evaluation examines the usability, effectiveness, and practical relevance of the AI-driven digital forensics framework, as perceived by digital forensic practitioners from the Center for Digital Forensics (CDF). The evaluation draws on feedback collected through a Google Form, featuring both rating scales and open-ended responses. The goal is to assess how well the framework achieves its objectives of simplifying forensic tasks, reducing technical expertise requirements, and improving investigation efficiency.

#### 4.1.2 Methodology

Feedback was gathered from two CDF practitioners using a structured Google Form. The form covered aspects such as overall satisfaction, UI/UX, task breakdown effectiveness, accuracy of analysis, clarity of AI agent guidance, code generation and execution, adoption and impact, and suggestions for additional features. Responses were analyzed using thematic analysis to identify key themes related to usability, clarity, and practical utility.

#### 4.1.3 Findings

##### 4.1.3.1 Usability and User Experience

The practitioners rated the UI/UX highly, assigning scores of 5 and 4 out of 5. Feedback indicated general satisfaction with the interface, though one practitioner highlighted the need for enhanced case management functionality.

#### **4.1.3.2 Effectiveness in Task Breakdown**

Both practitioners gave the tool a 4 out of 5 for its ability to break down complex forensic tasks into smaller, actionable steps. This suggests that the Task Translation Agent effectively simplifies forensic workflows, aligning with the framework’s objective of streamlining processes.

#### **4.1.3.3 Accuracy of Analysis**

The accuracy of the tool’s outputs received ratings of 3 and 4 out of 5. One practitioner emphasized the need for “comprehensive logging” to improve traceability and auditability, suggesting that greater transparency could bolster confidence in the tool’s analytical results.

#### **4.1.3.4 Clarity of AI Agent Guidance**

The clarity of the AI agents’ explanations and recommendations earned a consistent 4 out of 5 from both respondents. This indicates that the ReAct prompting method successfully enhances the understandability of the agents’ reasoning, supporting the objective of reducing technical expertise demands.

#### **4.1.3.5 Code Generation and Execution**

Satisfaction with the code generation and execution process was rated at 3 and 4 out of 5. While the framework’s ability to produce and run forensic scripts is a valuable feature, the mixed ratings suggest potential areas for improvement, such as script accuracy or execution efficiency.

#### **4.1.3.6 Adoption and Impact**

Both practitioners rated their likelihood of adopting the tool into their workflows at 5 out of 5, signaling strong practical relevance. One commented, “Good and beneficial tool for AI-assisted forensic investigations,” underscoring its perceived value in accelerating forensic processes.

#### **4.1.3.7 Suggestions for Improvement**

The practitioners offered actionable suggestions, including “Logs needs to be implemented” and “Implement comprehensive logging in the forensic tool to ensure all operations are traceable and auditable.” Additionally, one proposed integrating the tool with “Velocity” to further enhance its capabilities, providing clear directions for future development.

### **4.1.4 Conclusion**

The feedback from digital forensic practitioners highlights the framework’s strengths, such as its intuitive interface and capability to streamline complex forensic tasks, aligning closely with the design goals of improving usability and efficiency. Practitioners awarded perfect

adoption scores (5 out of 5), affirming the framework’s practical relevance and potential to address real-world forensic challenges. However, the evaluation also reveals opportunities for improvement, including the need for better logging, enhanced case management within the UI, and increased accuracy in code generation and execution. These findings validate the framework’s contributions to digital forensics while offering clear guidance for further refinement.

In light of these insights, the AI-driven digital forensics framework shows considerable promise as a tool for boosting investigation efficiency and lowering technical barriers. The positive practitioner feedback, combined with constructive suggestions for improved logging, advanced case management features, and integration with tools like "Velocity," charts a robust path for future development. Moving forward, efforts will concentrate on addressing these areas to enhance the framework’s utility and ensure it continues to meet the evolving demands of digital forensics practitioners.

## **4.2 Quantitative Evaluation**

### **4.2.1 Introduction**

The quantitative evaluation assesses the framework’s performance using AutoDFBench, an automated benchmarking tool designed to test AI-generated digital forensic tools against NIST’s Computer Forensics Tool Testing Program (CFTT) test procedures. This evaluation focuses on forensic string search tasks, measuring precision, recall, and F1 scores to quantify the framework’s accuracy and effectiveness compared to baseline LLM performance reported in the AutoDFBench study [12].

### **4.2.2 Integration of AutoDFBench with the AI Agent Framework**

To carry out the evaluation, AutoDFBench was integrated into our custom-built AI agent framework. This framework is designed to simulate the process of a digital forensic practitioner by coordinating multiple specialized agents. These include a Task Translation Agent, a Code Writing Agent, a Code Executor Agent, and a Reporter Agent. Each agent contributes to a specific part of the forensic workflow. The Task Translation Agent interprets the given forensic task based on a test case prompt, which is then handed over to the Code Writing Agent to generate an appropriate script—either in Python or shell—tailored to the task. The Code Executor Agent runs the script in an isolated environment, and finally, the Reporter Agent reviews the results and prepares the output.

In our implementation, we employed the LLAMA 3.3 70B language model with human feedback turned off to ensure a fully autonomous code generation process. The AI agent framework was designed to iteratively generate, evaluate, and refine its own code outputs without external human intervention. By allowing the agents to loop through cycles of planning, execution, and self-correction, the system demonstrated the capacity

to autonomously improve its solutions over time, showcasing the potential of large language models in fully automated forensic investigation workflows.

AutoDFBench, which was developed to benchmark forensic tools against standardized test cases, was a suitable fit for evaluating our framework. It supports forensic tasks performed via Python and shell scripts and aligns well with the script-oriented output generated by our agents. Additionally, our framework incorporates Sleuth Kit commands as part of the code generation process to retrieve file system-level information during analysis.

However, due to the current limitations of the AutoDFBench evaluation framework—specifically the lack of an integrated API—automated submission of results is not supported. As a result, after the agents generate and execute their scripts, the resulting data must be manually transferred into the AutoDFBench MySQL database. This involves manually writing SQL queries to insert the outputs into the `test_results` and `prompt_codes` tables. Although this step adds some overhead, it ensures that the results from our AI-driven framework can be fairly and consistently compared using AutoDFBench’s established scoring methods.

This integration made it possible to evaluate each test case in a structured and repeatable way. By using standard precision, recall, and F1 score metrics, we were able to quantify the performance of our AI agent system and benchmark it against baseline results from the original AutoDFBench study. Despite the manual step in the data flow, the integration was effective and aligned our experimental setup with widely accepted forensic evaluation standards.

### 4.2.3 Methodology

The AI-driven digital forensics framework was evaluated using a subset of string search test cases defined by the NIST Computer Forensics Tool Testing Program (CFTT), leveraging the AutoDFBench evaluation tool. The evaluation was conducted on NIST-provided disk images, which are part of the official NIST/CFTT String Search Data Set Version 1.1. This dataset includes two disk image files:

- **ss-win-07-25-18.dd** – A Windows-based disk image containing FAT, exFAT, and NTFS partitions along with unallocated space.
- **ss-unix-07-25-18.dd** – A Unix-based disk image including ext4, HFS+, and APFS file systems.

Each test case in the dataset is represented by multiple files containing strings that appear in specific conditions — such as live (active), deleted, or unallocated files — and encoded in various formats (ASCII, UTF-8, UTF-16BE, UTF-16LE). For example, for each base string, at least three string instances are created per test case: one in each of the active, deleted, and unallocated regions.

Across all test cases, the dataset includes hundreds of string instances, each embedded in a uniquely structured file, identified by a string ID and surrounded by nautical-themed filler text.

AutoDFBench reads these image files, defines the expected hits per test case, and compares them with the actual outputs of the forensic framework. For our evaluation, the focus was placed on the following string search tasks. AutoDFBench was configured to execute the following test cases, each representing a distinct forensic string search task:

- **FT-SS-01:** Search for an exact ASCII keyword  
*Example: Search for the exact word “WOLF” (case-sensitive match)*
- **FT-SS-02:** Search for an ASCII keyword ignoring case  
*Example: Match “wolf”, “WOLF”, or “WoLf” (case-insensitive)*
- **FT-SS-03:** Search only for whole words without substrings  
*Example: Match “WOLF” but not “WOLFPACK” or “WOLFLIKE”*
- **FT-SS-04:** Search with logical AND  
*Example: Find files containing both “DireWolf” and “WereWolf”*
- **FT-SS-05:** Search with logical OR  
*Example: Find files containing either “Dire” or “Were”*
- **FT-SS-06:** Search with logical NOT  
*Example: Find files containing “fox” but not “tiger”*
- **FT-SS-07-Norm:** Search Unicode strings for normalization forms and ligatures  
*Example: Match “mañana” whether encoded in NFC or NFD Unicode forms*
- **FT-SS-07-RTL:** Search Unicode written right-to-left (e.g., Arabic)  
*Example: Match the Arabic string “” regardless of directionality*
- **FT-SS-09-Stem:** Perform stemming search  
*Example: Match variations like “knife”, “knives”, “knifing” (same root word)*
- **FT-SS-10-Hex:** Search strings with hexadecimal characters  
*Example: Match the hex string `\x70\x61\x6E\x64\x61` which represents “panda”*

For each test case, the framework generated TSK commands via its AI agents, executed them on the disk image, and compared the outputs against NIST ground truth data. Metrics calculated included:

The evaluation metrics used in this study—**precision**, **recall**, and **F1 score**—were calculated using AutoDFBench’s built-in scoring mechanism. These metrics offer a quantitative perspective on how accurately the AI agent framework identified target strings within the disk image, compared against the expected results defined by the NIST CFTT test procedures.



- **True Positives (TP)**: Correct matches where the AI agents successfully detected a string in the correct file and storage location (e.g., active, deleted, or unallocated), as defined in the ground truth.
- **False Positives (FP)**: Incorrect matches where the agent reported a string that was not present in the ground truth.
- **False Negatives (FN)**: Relevant matches that were missed by the AI agents—instances where the string was expected but not detected.

Based on these values, the metrics are defined as follows:

- **Precision** =  $\frac{TP}{TP+FP}$ : Measures how many of the matches found by the agent were actually correct.
- **Recall** =  $\frac{TP}{TP+FN}$ : Measures how many of the expected matches were successfully detected by the agent.
- **F1 Score** =  $\frac{2\text{PrecisionRecall}}{\text{Precision}+\text{Recall}}$ : Provides a harmonic mean of precision and recall to give a balanced measure of accuracy.

In our framework, these metrics were automatically calculated by executing AutoDFBench’s `Summary.py` script after inserting the AI agent’s output into the `test_results` table and ensuring alignment with the corresponding ground truth entries. Due to computational limitations, each test case was executed only once, and the outcomes were logged in a spreadsheet titled for analysis. While the original AutoDFBench study averages results over multiple runs, our evaluation reflects performance based on a single run per test case due to resources restriction.

#### 4.2.4 Results

The performance metrics for each test case are summarized in Table 4.1.

Table 4.1: Performance Metrics for Forensic String Search Test Cases

Test Case	TP	FP	FN	Precision	Recall	F1 Score
FT-SS-01	4	3	3	0.5714	0.5714	0.5714
FT-SS-02	3	1	27	0.7500	0.1000	0.1765
FT-SS-03	8	0	12	1.0000	0.4000	0.5714
FT-SS-04	0	14	6	0.0000	0.0000	0.0000
FT-SS-05	0	14	12	0.0000	0.0000	0.0000
FT-SS-06	12	0	12	1.0000	0.5000	0.6667
FT-SS-07-Norm	7	0	15	1.0000	0.3182	0.4828
FT-SS-07-RTL	7	0	12	1.0000	0.3684	0.5385
FT-SS-09-Stem	4	3	93	0.5714	0.0412	0.0769
FT-SS-10-Hex	4	3	3	0.5714	0.5714	0.5714
<b>Average</b>	—	—	—	<b>0.6474</b>	<b>0.2870</b>	<b>0.3656</b>

**Note:** TP = True Positives, FP = False Positives, FN = False Negatives.

To provide a consolidated view of the framework’s performance across all test cases, a confusion matrix is presented in Table 4.2. The matrix aggregates the true positives, false positives, false negatives, and true negatives across all test cases.

Table 4.2: Confusion Matrix for Forensic String Search Test Cases

		<b>Predicted</b>	
		Positive	Negative
<b>Actual</b>	Positive	49	195
	Negative	38	0

**Note:** True Negatives (TN):number of places where the tool correctly did not find a string are not counted as ground truth only defines where strings should be found, not everywhere else they could possibly appear

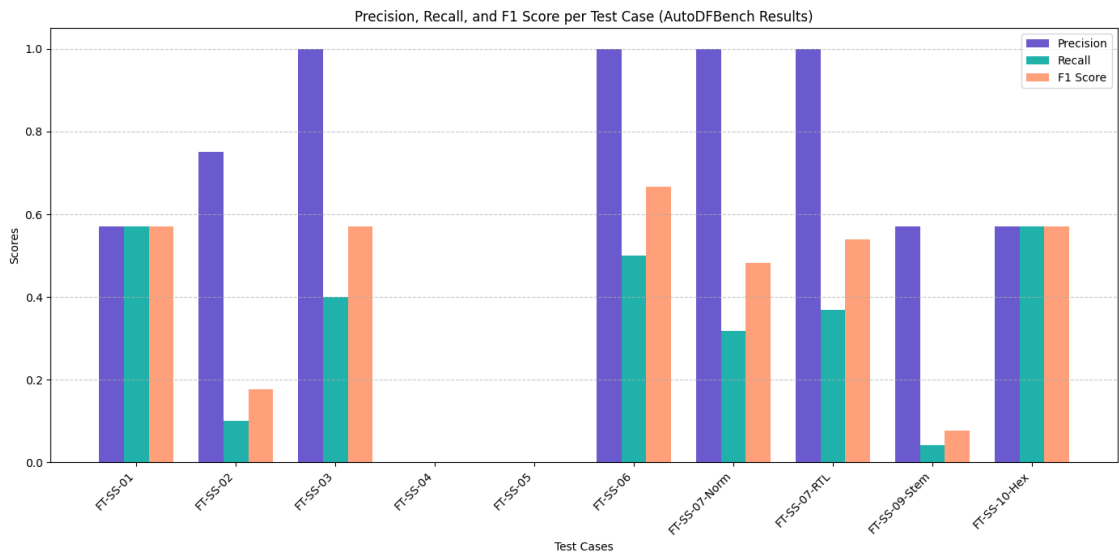


Figure 4.1: grouped bar chart with precision and recall

The grouped bar chart shows a deeper performance breakdown. High precision scores are observed for multiple test cases (especially FT-SS-03, FT-SS-06, FT-SS-07-Norm, and FT-SS-07-RTL), meaning that when predictions were made, they were often correct. However, recall values are much lower in several cases (e.g., FT-SS-02, FT-SS-09-Stem), indicating the framework missed many relevant results. This trade-off between precision and recall is reflected in the F1 Scores.

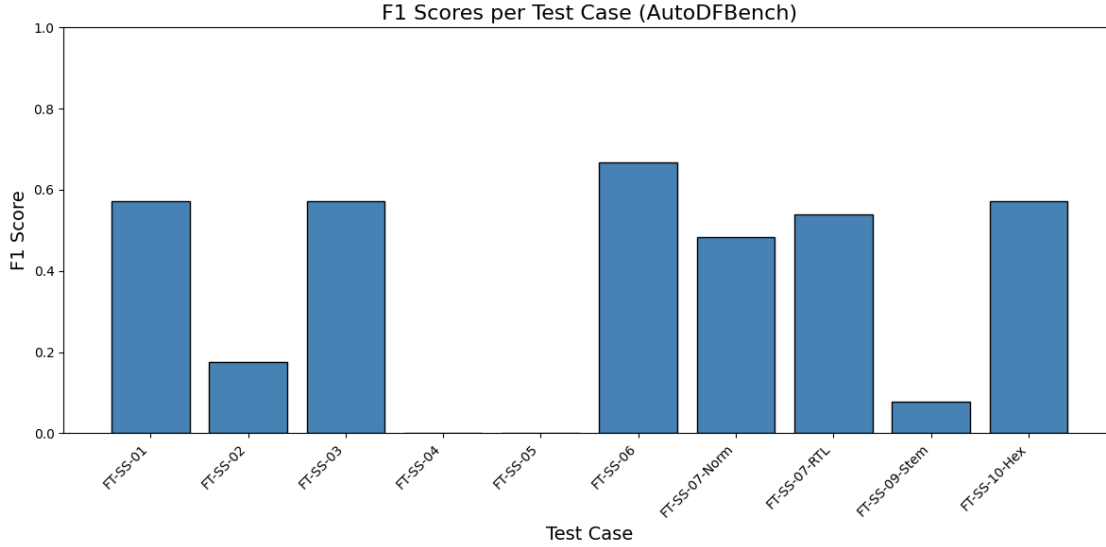


Figure 4.2: F1 score chart with test cases

The F1 Score analysis highlights significant variability across test cases. Certain test cases like FT-SS-06, FT-SS-07-Norm, and FT-SS-07-RTL show relatively stronger F1 performance (above 0.5), indicating successful string search task execution. However, some cases like FT-SS-04 and FT-SS-05 resulted in an F1 Score of 0, demonstrating challenges in handling specific string patterns or forensic complexities. The average F1 Score across all tests is approximately 0.366, suggesting that while the framework is capable, optimization opportunities remain for broader robustness. The framework excelled in tasks like FT-SS-06 (logical NOT,  $F1 = 0.6667$ ) and FT-SS-01, FT-SS-03, and FT-SS-10-Hex ( $F1 = 0.5714$  each), but struggled with FT-SS-04 and FT-SS-05 (logical AND/OR,  $F1 = 0$ ) and FT-SS-09-Stem (stemming,  $F1 = 0.077$ ).

## 4.2.5 Discussion

The quantitative results reveal a mixed performance profile. High F1 scores in tasks like FT-SS-06 (0.6667), FT-SS-01 (0.5714), FT-SS-03 (0.5714), and FT-SS-10-Hex (0.5714) demonstrate the framework’s capability for straightforward string searches, such as exact keyword, whole word, logical NOT, and hexadecimal searches. Precision values of 1.0 in several cases (e.g., FT-SS-03, FT-SS-06, FT-SS-07-Norm, FT-SS-07-RTL) indicate that detected instances were consistently correct, though lower recall suggests missed instances.

Conversely, the framework failed entirely on logical AND/OR tasks (FT-SS-04, FT-SS-05,  $F1 = 0$ ), likely due to difficulties in interpreting and executing complex logical queries. The stemming task (FT-SS-09-Stem,  $F1 = 0.077$ ) also showed poor performance, reflecting limitations in handling linguistic variations—possibly due to inadequate training of the LLM for such specialized tasks.

Compared to the AutoDFBench baseline, where top-performing LLMs (GPT-4o and Claude 3.5 Sonnet) achieved average F1 scores of 0.043 and 0.036 [12], our framework demonstrates a notable enhancement with an average F1 score of 0.366. However, it is

important to note that a direct comparison is not fully fair, as baseline models operated in a zero-shot setting without iterative self-correction, while our framework allows AI agents to iteratively refine and repair generated commands based on feedback. This iterative capability, combined with the integration of Retrieval-Augmented Generation (RAG), tool calling, and a multi-agent architecture, significantly boosts command accuracy and task success. Nevertheless, variability across test cases (F1 ranging from 0 to 0.6667) indicates that further improvements are needed to ensure more consistent performance.

The quantitative evaluation confirms the framework’s potential to outperform baseline LLMs in forensic string search tasks, with an average F1 score of 0.366. It excels in simpler searches but requires improvements for complex logical and linguistic tasks. These findings complement the qualitative feedback, reinforcing the need for enhanced reasoning and specialized training to achieve broader reliability.

### 4.3 Conclusion of Evaluation

The combined qualitative and quantitative evaluations demonstrate that the AI-driven digital forensics framework is a promising tool for enhancing investigation efficiency and accessibility. Practitioners value its usability and adoption potential, while AutoDFBench results show improved performance over standalone LLMs. However, challenges in handling complex queries and ensuring consistent accuracy across all tasks indicate areas for future work, such as fine-tuning the LLM on forensic-specific data and optimizing logical reasoning capabilities. Together, these results validate the framework’s contributions and provide a roadmap for its evolution into a robust forensic assistant.

# Chapter 5

## Discussion and Conclusion

### 5.1 Research Findings

The implementation of the digital forensic investigation framework proposed by [1] aimed to provide a robust tool for digital forensic practitioners to enhance the efficiency and accuracy of their investigative processes. During the initial phase of development, the framework utilized LLaMA 3, a large language model (LLM), to perform critical tasks such as generating a list of SleuthKit commands, which are fundamental to forensic analysis. When prompted with the query, "List the SleuthKit commands you know," the model produced a list that included several inaccuracies, a phenomenon known as hallucination, where LLMs generate plausible but incorrect information due to limitations in their internal knowledge [1][9]. This issue threatened the framework's reliability for practitioners who depend on precise tools. To address this, we incorporated a Retrieval-Augmented Generation (RAG) approach, integrating official SleuthKit documentation into the LLM's context window. By grounding the model's outputs in verified external data, this method significantly reduced hallucination, ensuring that the generated command list was accurate and trustworthy. This approach is consistent with findings by [9], who demonstrated that leveraging external knowledge improves the factual accuracy of LLM outputs, thereby enabling forensic practitioners to rely on the framework for precise and dependable task execution.

Another significant challenge was ensuring that the framework supported the logical reasoning process required by digital forensic reporting, particularly in generating clear and comprehensive reports that reflect the investigative workflow. Initial attempts using standard prompting techniques produced outputs that were disjointed and lacked the coherence necessary for professional forensic reporting. To overcome this, we adopted the ReAct prompting method, which interleaves reasoning and action[9] steps within the LLM's context window. This approach was applied to the Task Translation Agent, which breaks down complex forensic tasks into manageable subtasks, and the Coder Agent, which generates scripts to facilitate analysis. By maintaining a record of the agents' thought processes and action plans, the framework ensured a structured and transparent workflow, enabling the Reporter Agent to produce detailed, logically coherent reports that aligned with the needs of forensic practitioners. Following guidance to focus the tool exclusively on trained digital

forensic professionals, the framework was tailored to support their specialized expertise, enhancing their ability to conduct thorough investigations while maintaining the precision and clarity required for professional forensic applications.

Primary experiments with the framework revealed an additional challenge related to the management of large context windows within the Autogen framework, which impacted the tool’s performance for digital forensic practitioners. In the Autogen framework, it stores information generated by all agents, such as conversations of the agents. When a large context message was included as a system message for an agent, and the results were passed between agents, the framework became increasingly populated, resulting in a substantial volume of content being fed into the LLM for processing. Our observations indicated that as the context window grew larger, the framework began to act abruptly, producing inconsistent or erratic outputs. This issue, noted during initial testing, highlights a critical limitation in handling extensive context within LLM-based agent frameworks, as excessive input can overwhelm the model’s capacity to maintain coherence and accuracy [1]. Addressing this challenge will require strategies to optimize context management, such as truncating irrelevant information or prioritizing critical data which is send to LLM, to ensure the framework remains stable and effective for forensic practitioners conducting complex investigations.

To address hallucination in the digital forensic investigation framework and enhance its reliability for practitioners, we introduced a tool-calling feature that retrieves accurate command information from the official SleuthKit documentation on the web, supplementing the LLM’s knowledge when deficient, as outlined by[1]. This approach effectively reduced inaccuracies in generated outputs by incorporating real-time, verified SleuthKit data. However, the addition of this data intensified context window overload within the Autogen framework, as the messages variable accumulated large volumes of information, leading to abrupt and inconsistent performance in primary experiments. This issue corroborates findings by [21], who demonstrated that LLMs exhibit degraded performance with larger contexts, particularly failing to effectively process information in the middle of extended context windows, resulting in reduced coherence and accuracy. To mitigate this, we refined the tool-calling mechanism by employing two agents in a nested chat structure to extract and filter the most relevant web-based information, delivering it to the main framework in a concise JSON format. This optimization significantly reduces context window size while preserving critical data integrity, aligning with [9]’s findings on efficient external data integration, thereby ensuring the framework’s stability and effectiveness for digital forensic practitioners conducting complex investigations.

In the initial stages of developing the digital forensic investigation framework, resource constraints necessitated the use of 7-billion parameter quantized models, which provided a feasible starting point for testing and refining the framework’s capabilities for digital forensic practitioners. These smaller models, while computationally efficient, exhibited limitations in handling complex tasks and maintaining output accuracy, particularly under the demands of

forensic analysis, as noted by [1]). Toward the conclusion of the research, access to powerful servers enabled the deployment of larger 20-billion and 30-billion parameter models, which markedly enhanced the framework’s performance. These advanced models demonstrated superior contextual understanding and reduced hallucination, leading to more accurate generation of SleuthKit commands and coherent investigative reports, aligning with findings by [9] on the benefits of scaling model size for specialized applications. The transition to higher-capacity models significantly improved the framework’s reliability and efficiency, underscoring the importance of computational resources in optimizing LLM-based tools for professional digital forensic investigations.

## 5.2 Discussion

This research was guided by two primary questions: to what extent can an AI-assisted digital forensic framework enhance the efficiency of practitioners, and how effectively can such a system be deployed as a practical, user-friendly tool for real-world forensic investigations. The evaluation—consisting of both practitioner-based qualitative feedback and quantitative benchmarking using AutoDFBench—yields substantive insights into the performance and applicability of the proposed solution.

The first research question focuses on practitioner efficiency, particularly in time-critical scenarios where repetitive command-line tasks and data parsing hinder rapid forensic insight. The qualitative evaluation involving practitioners from the Center for Digital Forensics in UCSC revealed that the system assist in their workflows. Rather than replacing expert knowledge, the framework acts as an intelligent assistant, rapidly interpreting high-level forensic queries into executable SleuthKit commands, shell scripts and python scripts. This delegation of low-level operations allowed practitioners to concentrate more on investigative strategy and hypothesis formation, rather than spending time crafting or debugging commands. In essence, the AI agents within the system functioned as cognitive and procedural accelerators, increasing the throughput and clarity of forensic tasks without abstracting away the domain’s technical rigor.

Despite this advantage, certain limitations emerged that must be considered for real-world deployment. The practitioners highlighted issues such as command inconsistency in varying system environments, and the absence of features like integrated case management or error traceback. These shortcomings point to a broader need for robustness in forensic tooling—especially when used in legal or compliance-sensitive contexts. Nonetheless, these gaps are not insurmountable and do not detract from the system’s core strength: reducing task complexity and execution time, thereby improving practitioner efficiency across common investigative scenarios.

The second research question relates to the broader deployment potential of the tool. The quantitative evaluation conducted using AutoDFBench offered objective performance metrics that support the framework’s viability. Precision and recall values for straightforward string search tasks were high, with some test cases achieving perfect precision. These results confirm

the tool’s accuracy in basic evidence identification workflows. However, its effectiveness diminished when handling complex logical expressions, such as queries involving Boolean operators or semantic variations. This suggests that while the current system is optimized for clarity and speed, enhancements in language parsing and logical reasoning would be required to extend its utility to more nuanced forensic searches.

Practitioner feedback corroborated these findings. The AI framework was noted to be user-friendly and time-efficient, particularly in reducing the friction associated with manual data review and command-line scripting. However, suggestions for case-specific organization, historical activity logging, and enhanced auditability underscore the importance of forensic transparency and traceability in professional settings. These insights provide a roadmap for future iterations of the tool, particularly in contexts where reproducibility and legal defensibility are paramount.

Moreover, the current evaluation setup—especially the manual process of inserting AutoDFBench results into a MySQL database—presents an operational bottleneck. Automating this integration in future development stages would not only streamline benchmarking but also allow for continuous validation and performance monitoring as the tool evolves. While the proposed AI-assisted digital forensic framework demonstrates promising capabilities in several cases, it is crucial to recognize that its performance is neither universally consistent nor sufficient to be deemed a foolproof solution for professional forensic investigations. This is particularly evident in test cases such as FT-SS-04 and FT-SS-05, where the framework’s output failed to meet the minimum acceptable standards, indicating a substantial shortfall in reliability. These failures highlight that the system cannot yet be considered a silver bullet for digital forensics, a domain in which precision and accuracy are not just desirable but essential due to the legal and procedural sensitivities involved. Furthermore, even in scenarios where the tool produced acceptable outcomes, the F1-scores across multiple other test cases were below what is typically expected of forensic-grade tools. Given the binary and deterministic nature of many digital forensic tasks, such as exact string matching or evidence path reconstruction, any fluctuation in accuracy—even when marginal—can have significant implications for the integrity of an investigation. The observed inconsistency in results, despite utilizing LLAMA 3.3-70B, a state-of-the-art large language model, suggests a deeper limitation within current LLM architectures when applied to the domain of digital forensics. Specifically, it reveals that general-purpose LLMs lack domain specialization in interpreting, reasoning, and executing tasks that require strict logical fidelity and forensic semantics. The performance variance implies that these models are not yet attuned to the precision demands inherent in digital evidence analysis, and as such, cannot be relied upon for critical forensic determinations without human oversight. This reinforces the necessity of domain-specific fine-tuning, where LLMs are trained on curated forensic data, structured tool outputs, and legal procedures to better understand and respond to the unique linguistic and procedural contexts of digital forensics. However, despite these limitations, the framework’s design strategically integrates



a human-in-the-loop approach, allowing digital forensic practitioners to intervene and validate or override AI-generated outputs. This design choice ensures that when the model encounters ambiguous or unfamiliar forensic contexts—such as those observed in poorly performing test cases—human experts can step in to maintain procedural accuracy and evidentiary integrity. Rather than replacing the role of practitioners, the framework enhances their efficiency by automating repetitive or time-consuming tasks while still preserving critical decision-making in the hands of qualified investigators. Thus, the integration of human oversight serves not only as a safeguard against AI error but also as a bridge between emerging AI capabilities and the rigorous demands of forensic practice. It provides a practical path forward where artificial intelligence functions as an assistive tool—one that augments, rather than undermines, the responsibilities of digital forensic experts.

### 5.3 Final Conclusion

This research sought to address two key questions: whether the integration of Large Language Models (LLMs) into digital forensic (DF) investigations can reduce the demand for technical expertise among practitioners, and how to effectively implement an AI-driven DF framework as a user-friendly tool for real-world forensic investigations. Through the design, development, and evaluation of a multi-agent framework leveraging LLMs, this study demonstrates how artificial intelligence can augment forensic workflows while maintaining critical human oversight.

In response to the first research question, our findings indicate that integrating LLMs significantly reduces the technical barriers traditionally associated with file system analysis and forensic tool usage. By incorporating Retrieval-Augmented Generation (RAG), ReAct prompting, and an iterative self-correction mechanism, the framework translates natural language queries into executable forensic commands without requiring users to manually craft or troubleshoot complex command-line instructions. Quantitative evaluation using AutoDFBench revealed an average F1 score of 0.366 across a variety of forensic string search tasks, representing a substantial improvement compared to baseline zero-shot LLM performance (0.043 for GPT-4o and 0.036 for Claude 3.5 Sonnet). While the performance varied across test cases—with F1 scores ranging from 0.0 to 0.6667—the results underscore the framework’s ability to automate routine forensic tasks while minimizing errors associated with manual command generation. This improvement can be attributed to the framework’s multi-agent architecture, which distributes responsibilities across specialized agents for task translation, code generation, execution, and reporting, effectively streamlining investigations and reducing cognitive load for practitioners.

Addressing the second research question, the framework was implemented with a strong emphasis on usability, modularity, and real-world applicability. A user-friendly chat interface was developed to allow investigators to interact with the system using natural language, while integrated logging and case management features ensured traceability and transparency—both of which are critical in legal and procedural contexts. Practitioner

feedback gathered through qualitative evaluation confirmed the system’s potential to improve investigation efficiency by reducing time-consuming manual steps and simplifying access to complex forensic tools like The Sleuth Kit (TSK). The incorporation of dynamic tool-calling functionality further enhanced the framework’s adaptability, allowing it to access relevant documentation in real-time to improve command accuracy and mitigate LLM hallucinations.

However, the evaluation also revealed limitations that necessitate human oversight. While the iterative self-correction process enabled the framework to identify and fix execution errors autonomously, performance variability across test cases highlighted the need for continuous human supervision. In sensitive investigative environments, where precision and recall are critical, practitioners must remain actively involved in validating AI-generated outputs and ensuring that investigative processes align with forensic standards. The integration of a human-in-the-loop approach serves this purpose by allowing practitioners to review, approve, or override AI-generated commands when necessary, thereby preventing erroneous results from compromising the integrity of an investigation. This supervision is particularly important in cases where the framework struggled, such as logical AND/OR searches (FT-SS-04 and FT-SS-05) or stemming tasks (FT-SS-09-Stem), where poor recall and F1 scores revealed limitations in the LLM’s domain-specific reasoning capabilities. Beyond quantitative performance gains, the qualitative evaluation further validates the framework’s practical value. Feedback from digital forensic practitioners emphasized strong satisfaction with the system’s usability, particularly highlighting its intuitive user interface and effective task breakdown mechanisms. High adoption scores (5 out of 5) indicate a strong likelihood that practitioners would integrate the tool into real-world forensic workflows, underscoring the framework’s readiness for operational use.

Practitioners also recognized the effectiveness of the AI agents in simplifying complex forensic procedures, with consistent ratings of 4 out of 5 for task breakdown clarity and AI agent guidance. These results align with the framework’s primary goal of reducing the technical expertise typically required for advanced forensic analysis, suggesting that the system successfully bridges the gap between technical complexity and practical usability.

However, the evaluation also surfaced critical areas for further refinement. Specifically, practitioners highlighted the need for comprehensive logging mechanisms to improve traceability and auditability, along with suggestions for enhanced case management features within the UI. These insights provide a clear and actionable roadmap for future development phases, ensuring that the framework evolves to meet the operational standards expected in forensic environments.

The combination of promising quantitative results and positive practitioner feedback affirms the framework’s potential to democratize access to advanced forensic tools. By continuing to address user-suggested improvements—such as robust logging and possible integration with complementary platforms like Velocity—the framework is well-positioned to drive the next wave of AI-assisted innovation in digital forensics.

Overall, this research demonstrates that while LLM integration can significantly

reduce technical demands and improve the efficiency of digital forensic investigations, the framework’s effectiveness is maximized when combined with human validation and oversight. By striking a balance between automation and practitioner control, the proposed system enhances investigative workflows while safeguarding against the risks associated with AI-generated outputs. Future work will focus on further fine-tuning LLMs with forensic-specific data to address performance inconsistencies, expanding the framework’s support for additional forensic tasks, and refining human-in-the-loop mechanisms to ensure seamless collaboration between AI agents and practitioners.

Ultimately, this study contributes a novel, user-centric solution for digital forensics, paving the way for future AI-powered systems that maintain both operational efficiency and evidentiary integrity in increasingly complex investigative environments.

## 5.4 Recommendation

To further enhance the digital forensic investigation framework for practitioners, several strategies are recommended based on the challenges and observations encountered during its development. First, to address the issue of context window overload observed in the Autogen framework, which led to degraded performance as noted by [21], we recommend implementing dynamic context pruning mechanisms. These mechanisms would prioritize and retain only the most relevant information within the messages variable, such as recent agent outputs and critical SleuthKit documentation retrieved via the tool-calling feature, while discarding redundant or less pertinent data. This approach would align with [9]’s findings on optimizing external data integration and ensure that the LLM processes manageable context sizes, thereby improving output coherence and stability. Additionally, the refined tool-calling mechanism, utilizing two agents in a nested chat structure to deliver concise JSON-formatted data, should be further standardized and automated to streamline real-time data retrieval from sources like the SleuthKit documentation, as proposed by [1]. This would minimize hallucination while maintaining a lean context window, enhancing the framework’s reliability for forensic tasks.

## 5.5 Future Work

### 5.5.1 Multimodal Forensics Integration.

Currently, the framework is designed primarily for file system analysis using tools such as Sleuth Kit. However, in real-world forensic investigations, evidence is not limited to file metadata or directory structures. Investigators often need to analyze multimedia content such as images and videos, as well as communication data captured from network traffic. Therefore, extending the framework to support multimodal forensic capabilities is an important next step.

This can be achieved by integrating additional forensic tools that specialize in media file

carving, facial recognition, object detection in images, and packet analysis of network traffic. For instance, tools like *Volatility* for memory analysis or *Wireshark* for network capture interpretation could be incorporated alongside Sleuth Kit. By doing so, the framework would evolve into a more holistic digital forensic assistant capable of handling diverse forms of digital evidence. Such an expansion would not only make the system more robust but also align it better with the realities of complex cybercrime cases.

### 5.5.2 Support to Other Digital Forensic Branches

Modern investigations frequently involve mobile devices and cloud-based services, which often store critical digital traces related to communication, location history, and user behavior. As such, expanding the framework to include mobile and cloud forensics is crucial to its relevance and scalability.

This involves adapting the AI agents to interact with forensic tools like *Cellebrite UFED* or *Magnet AXIOM* for mobile device analysis, as well as integrating APIs that can retrieve and analyze data from cloud platforms such as Google Drive, iCloud, or Microsoft OneDrive. By incorporating these capabilities, the framework will offer broader coverage across various digital forensic branches, enabling investigators to conduct end-to-end analysis within a single unified environment. This extension is particularly important in scenarios involving cyberstalking, fraud, and data breaches, where data is often dispersed across multiple platforms.

### 5.5.3 Create a Dataset for Fine-Tuning an LLM

The current implementation of the framework utilizes general-purpose Large Language Models (LLMs), which, while capable, lack specialized training in digital forensic tasks. To improve the model's contextual understanding, relevance, and precision in forensic command generation, a domain-specific dataset must be developed for fine-tuning.

The creation of this dataset would involve the collection of forensic cases, sample command sequences, disk images, timelines, and incident reports that reflect realistic investigative scenarios. Importantly, the Center for Digital Forensics (CDF) lab has indicated its willingness to contribute anonymized data for this purpose. Using such data, a fine-tuned LLM could be trained to better understand the structure and semantics of digital forensic inquiries, including nuances such as interpreting timestamps, recognizing common artefact patterns, and differentiating between file system types. This would allow the model to perform more reliably across various levels of user expertise.

### 5.5.4 Framework Testing with Fine-Tuned LLMs

Once a fine-tuned forensic-specific LLM has been developed, the next logical step would be to integrate it into the existing framework and conduct a thorough evaluation of its performance. This testing phase would involve comparing the fine-tuned model's outputs with those of general-purpose LLMs using the same test scenarios and prompts.

Key performance metrics such as command accuracy, task completion time, error rate, and user satisfaction could be benchmarked using established evaluation tools like *AutoDF Bench*. In addition, practitioner feedback could be gathered to assess improvements in usability and interpretability. This evaluation would offer concrete evidence as to whether domain-specific training significantly enhances the practical applicability of LLMs in digital forensic contexts.

Ultimately, such fine-tuning and testing efforts would move the framework closer to becoming a deployable, expert-level assistant capable of supporting law enforcement agencies and digital forensic analysts in high-stakes investigations.

# References

- [1] A. Wickramasekara and M. Scanlon, “A framework for integrated digital forensic investigation employing autogen ai agents,” in *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 01–06, IEEE, 2024.
- [2] B. Carrier, *File System Forensic Analysis*. Addison-Wesley, 2005.
- [3] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, “Autogen: Enabling next-gen llm applications via multi-agent conversation framework,” *arXiv preprint arXiv:2308.08155*, 2023.
- [4] Amazon Web Services, Inc., “What are AI Agents? - Agents in Artificial Intelligence Explained.” <https://aws.amazon.com/what-is/ai-agents/>. Accessed: Apr. 27, 2025.
- [5] T. Hussain, T. Ahmed, M. S. HAQUE, and M. R. A. Rashid, “Collective wisdom in language models: Harnessing llm-swarm for agile project management,” in *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
- [6] C. Hymel, S. Peng, K. Xu, and C. Ranganathan, “Improving performance of commercially available ai products in a multi-agent configuration,” *arXiv preprint arXiv:2410.22129*, 2024.
- [7] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, “A systematic survey of prompt engineering in large language models: Techniques and applications,” *arXiv preprint arXiv:2402.07927*, 2024.
- [8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.
- [9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “React: Synergizing reasoning and acting in language models,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [10] S. L. Garfinkel, “Automating disk forensic processing with sleuthkit, xml and python,” in *2009 Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 73–84, 2009.

- [11] E.-P. D. Kakalis and S.-A. Kefalidis, “Advancing geosparql query generation on yago2geo: Leveraging large language models and automated uri injection from natural language questions,” 2024.
- [12] A. Wickramasekara, A. Densmore, F. Breitingner, H. Studiawan, and M. Scanlon, “Autodfbench: A framework for ai generated digital forensic code and tool testing and evaluation,” in *Proceedings of the Digital Forensics Doctoral Symposium*, pp. 1–7, 2025.
- [13] M. Scanlon, F. Breitingner, C. Hargreaves, J.-N. Hilgert, and J. Sheppard, “Chatgpt for digital forensic investigation: The good, the bad, and the unknown,” *Forensic Science International: Digital Investigation*, vol. 46, p. 301609, 2023.
- [14] A. Wickramasekara, F. Breitingner, and M. Scanlon, “Sok: Exploring the potential of large language models for improving digital forensic investigation efficiency,” *arXiv preprint arXiv:2402.19366*, 2024.
- [15] C. Hargreaves, F. Breitingner, L. Dowthwaite, H. Webb, and M. Scanlon, “DFPulse: The 2024 Digital Forensic Practitioner Survey,” Sept. 2024.
- [16] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [17] Streamlit, “A faster way to build and share data apps.” <https://streamlit.io/>. Accessed: Apr. 27, 2025.
- [18] Microsoft, “AutoGen.” <https://microsoft.github.io/autogen/0.4.0/index.html>. Version 0.4.0; Accessed: Apr. 27, 2025.
- [19] Chainlit, “Build AI applications.” <https://chainlit.io/>. Accessed: Apr. 27, 2025.
- [20] N. I. of Standards and Technology, “CFReDS Portal.” <https://cfreds.nist.gov/>.
- [21] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, “Lost in the middle: How language models use long contexts,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.

# Appendices



# Appendix A

## Screenshots of the developed graphical user interface

This appendix provides screenshots of the graphical user interface developed as a means of making the framework a user-friendly DF tool. It also served as a tool for qualitative evaluation on what was needed by practitioners from a DF tool of this stature.

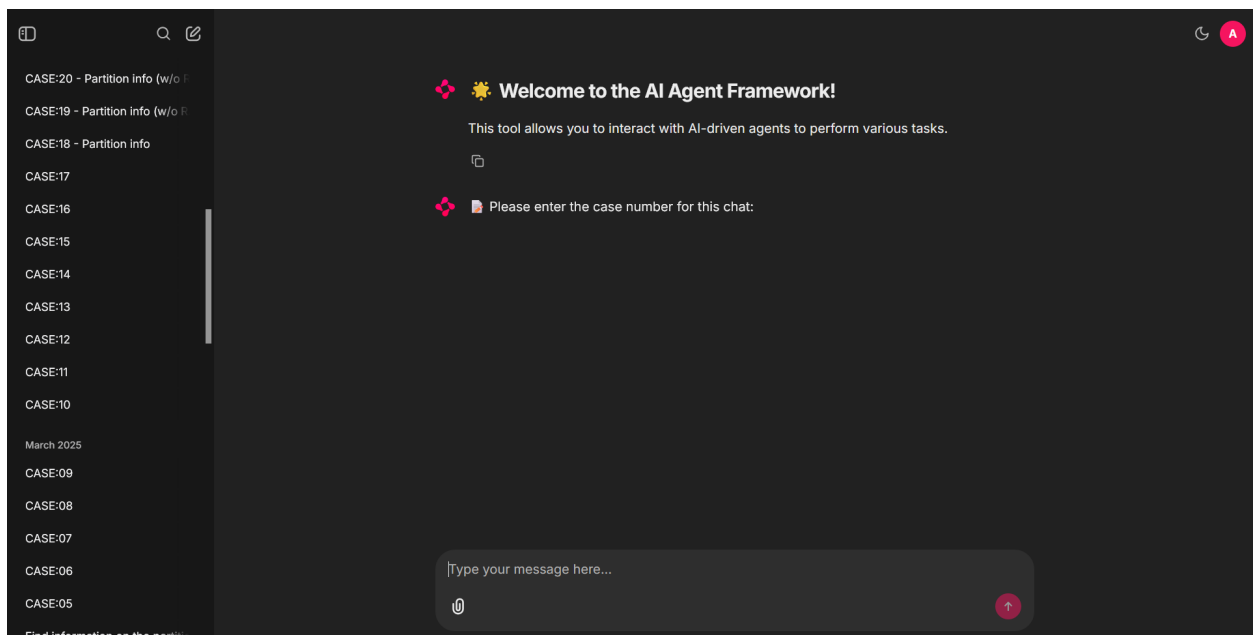


Figure 1.1: Full interface

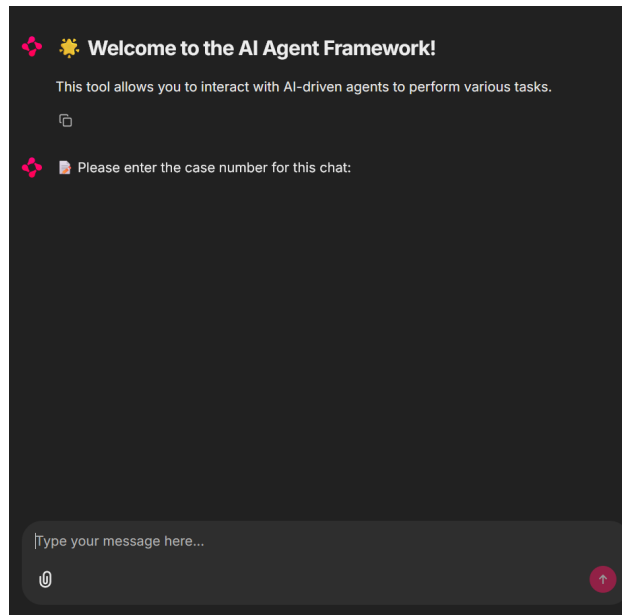
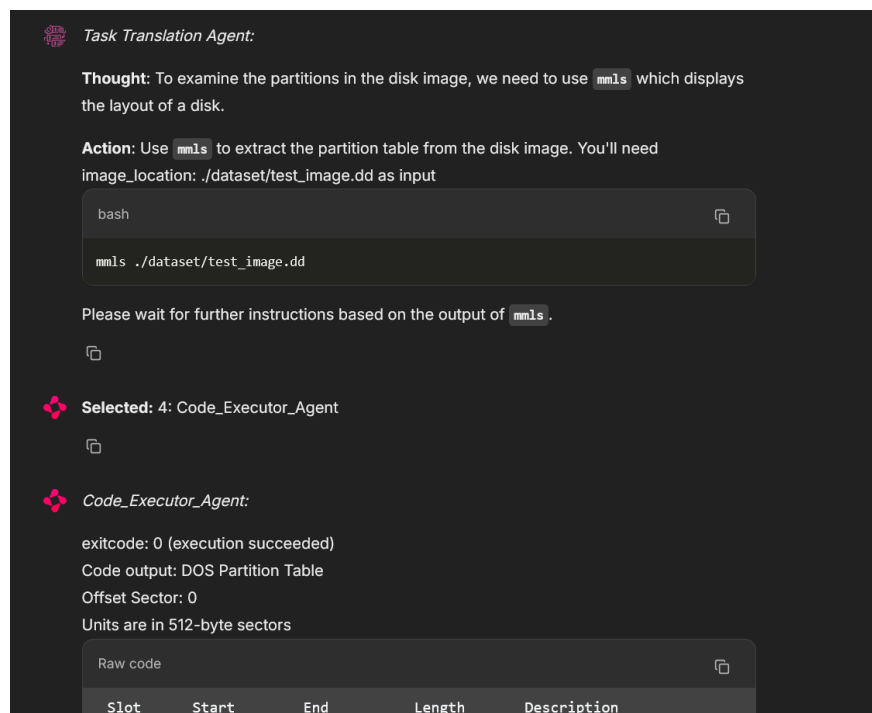


Figure 1.2: Opening chat interface for a chat requiring a case number



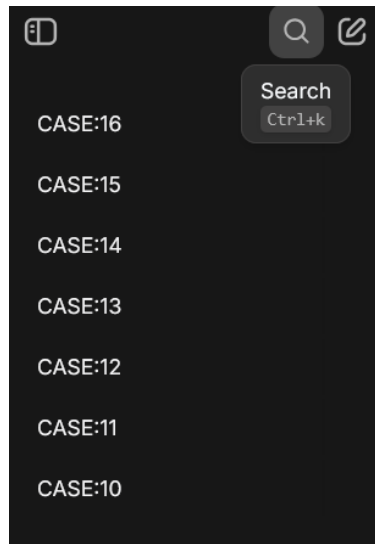


Figure 1.4: Side bar for tracking chats with a case number with the option to search

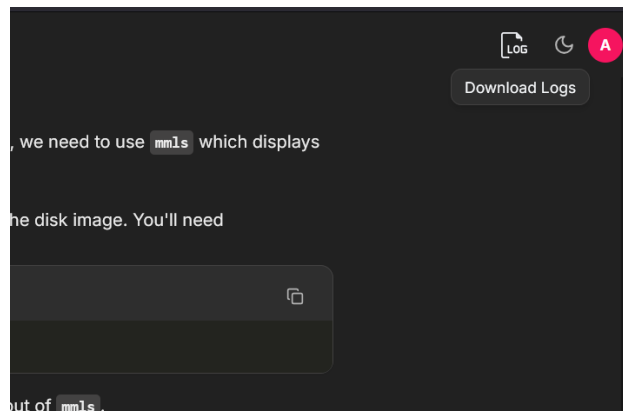


Figure 1.5: Button to download logs

```
2025-03-07 14:56:53 [INFO] chainlit_audit_logger: [CHAINLIT] Received user message: Find information on the partitions in the disk image
image_location: ./dataset/test_image.dd
2025-03-07 14:56:53 [INFO] chainlit_audit_logger: [CHAINLIT] Processing user message: Find information on the partitions in the disk image
image_location: ./dataset/test_image.dd
2025-03-07 14:56:54 [INFO] chainlit_audit_logger: [ChainlitGroupChatManager] Processing message from RAG_Proxy_Agent, silent=False
2025-03-07 14:56:54 [INFO] chainlit_audit_logger: [ChainlitGroupChatManager] Message content: You're a retrieve augmented chatbot. You answer user's questions based on your own knowledge and the
context provided by the user.
If you can't answer the question with or without the current context, you should reply exactly 'UPDATE CONTEXT'.
You must give as short an answer as possible.

User's question is: Find information on the partitions in the disk image
image_location: ./dataset/test_image.dd

Context is:
    partitions and to identify the file system offset for The Sleuth Kit tools. The media management tools support DOS
    partitions, BSD disk labels, Sun VTOC, and Mac partitions.</p>
<ul>
  <li><strong>mmls</strong>: Displays the layout of a disk, including the unallocated spaces.</li>
  <li><strong>mmstat</strong>: Display details about a volume system (typically only the type).</li>
  <li><strong>mmcat</strong>: Extracts the contents of a specific volume to STDOUT.</li>
</ul>
<h1 id="image-file-tools">Image File Tools</h1>
<p>This layer contains tools for the image file format. For example, if the image format is a split image or a
compressed image.</p>
```

Figure 1.6: Generated log file

# Appendix B

## Screenshots of Qualitative Evaluation Findings

This appendix provides the screenshots of the detailed feedback collected during the qualitative evaluation of the AI-driven digital forensics framework. These images correspond to the findings discussed in Section ?? and present practitioner ratings and comments for different evaluation criteria.

### 2.1 Overall Satisfaction

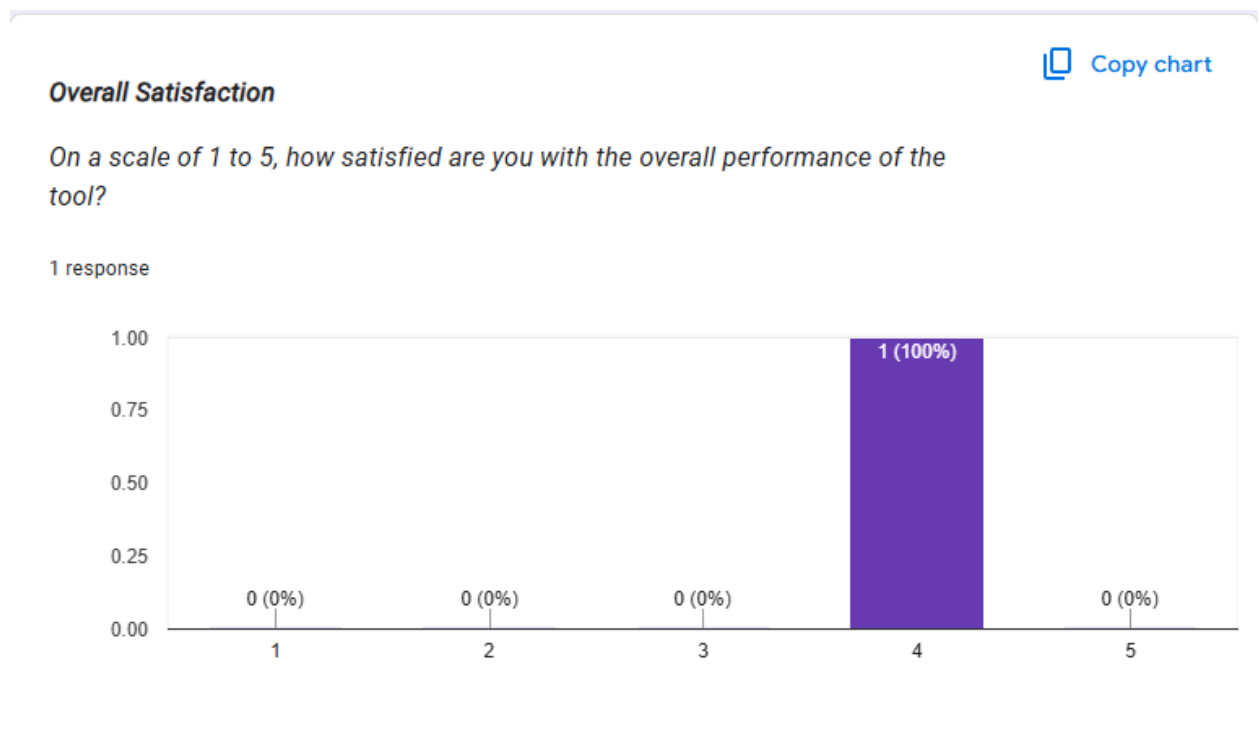


Figure 2.1: Practitioner feedback on Overall Satisfaction

## 2.2 UI/UX Evaluation

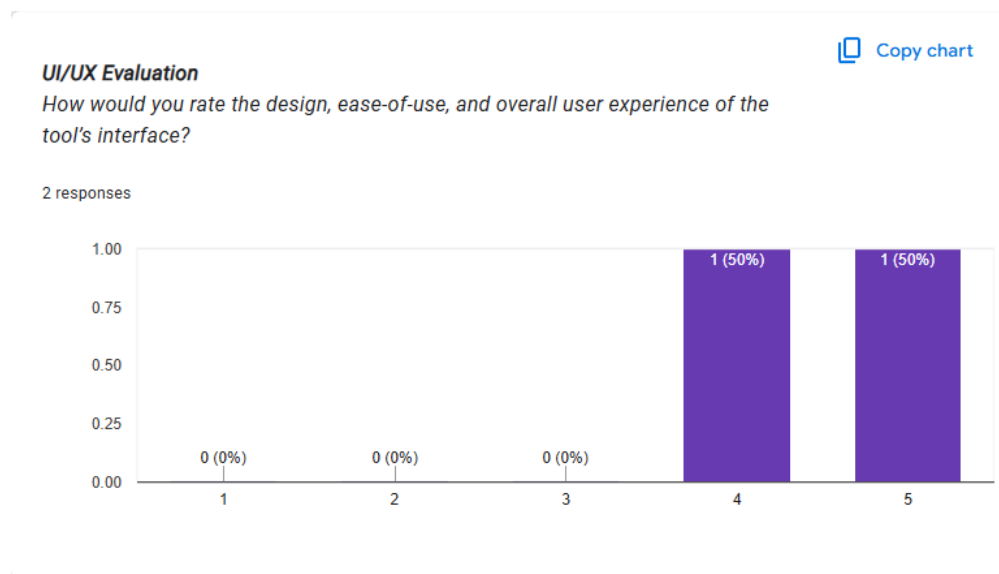


Figure 2.2: Practitioner feedback on UI/UX Evaluation

## 2.3 Task Breakdown Effectiveness

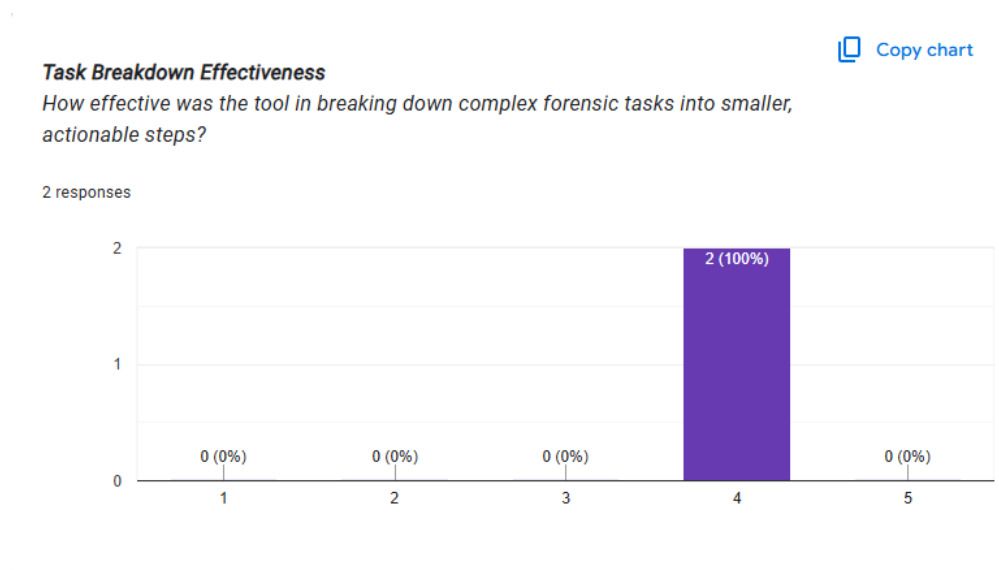


Figure 2.3: Practitioner feedback on Task Breakdown Effectiveness

## 2.4 Accuracy of Analysis

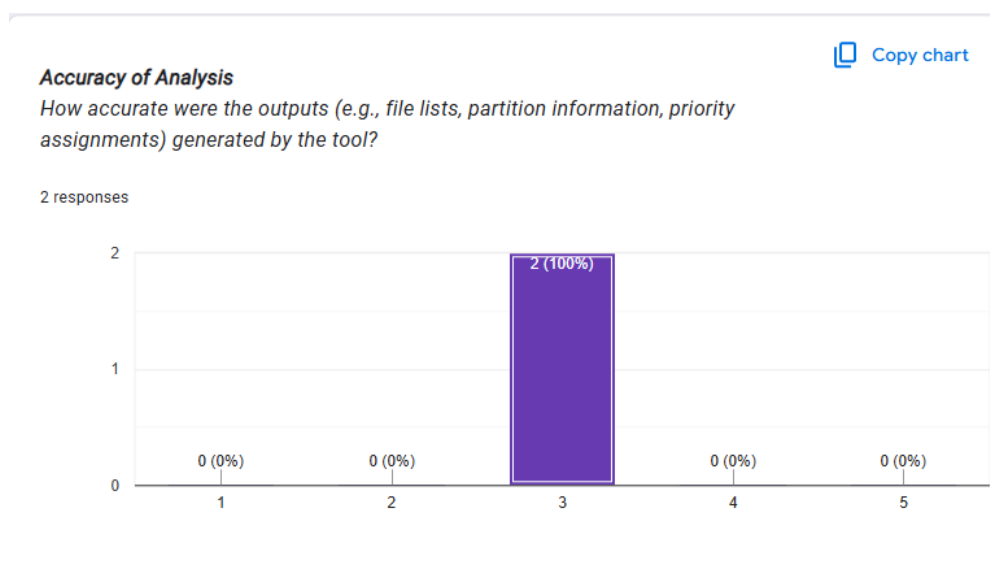


Figure 2.4: Practitioner feedback on Accuracy of Analysis

## 2.5 Clarity of AI Agent Guidance

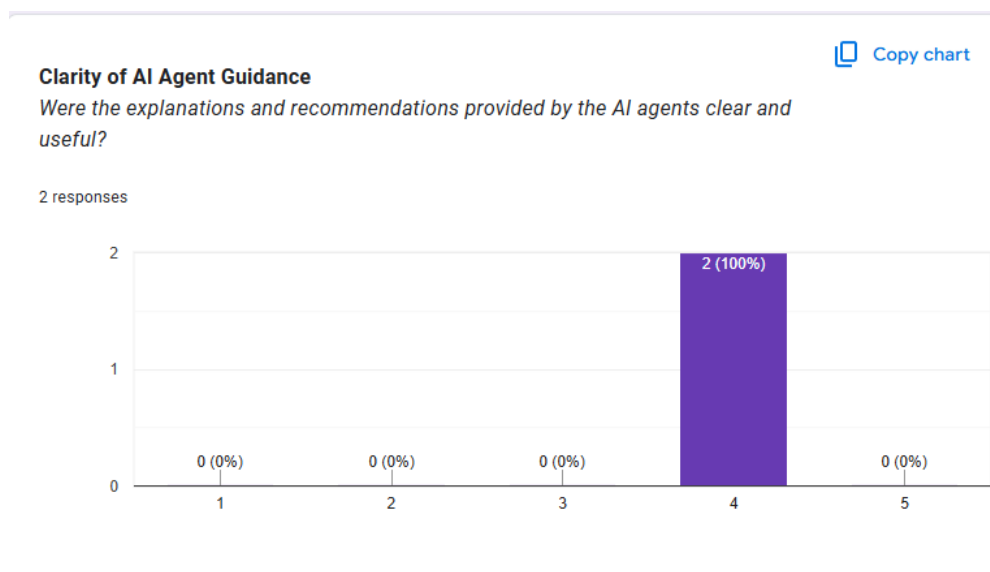


Figure 2.5: Practitioner feedback on Clarity of AI Agent Guidance

## 2.6 Code Generation & Execution

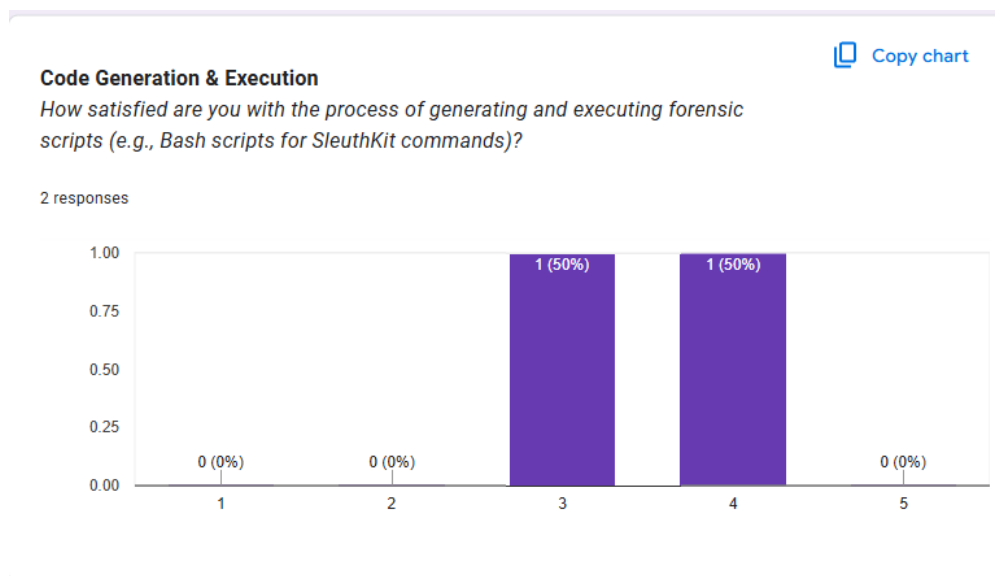


Figure 2.6: Practitioner feedback on Code Generation and Execution

## 2.7 Adoption and Impact

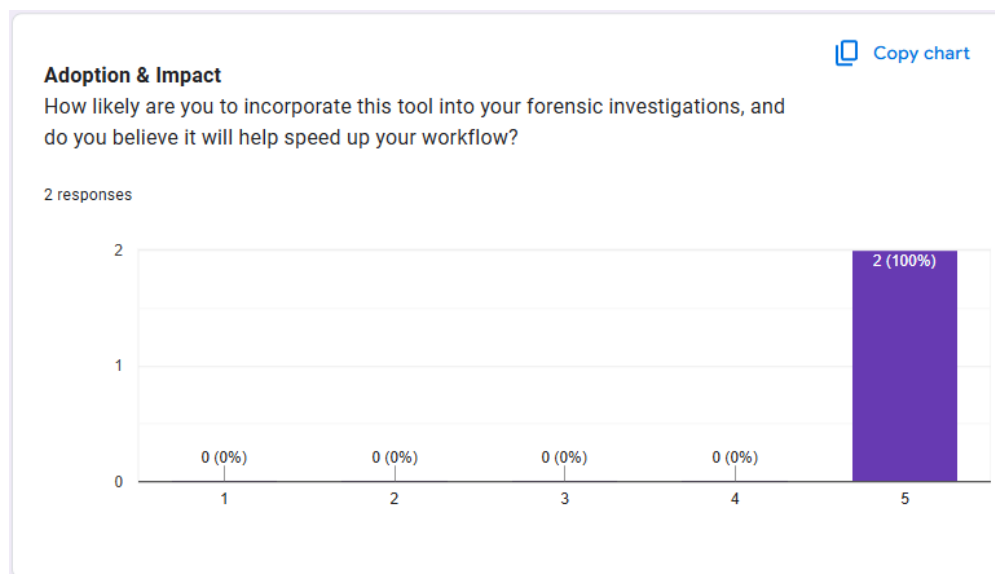


Figure 2.7: Practitioner feedback on Adoption and Impact

## 2.8 Additional Features

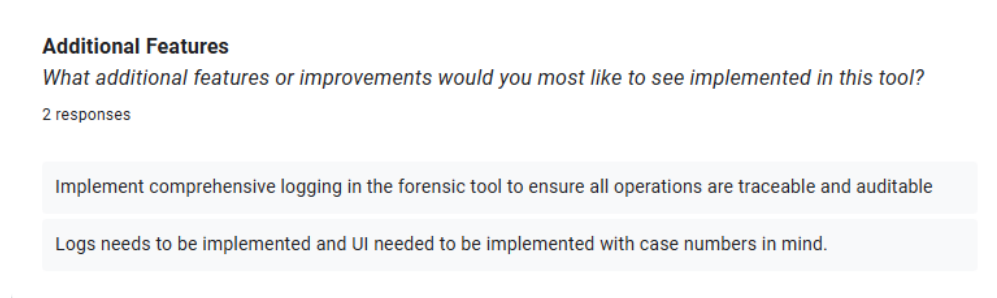


Figure 2.8: Practitioner requests for Additional Features

## 2.9 General Comments & Suggestions

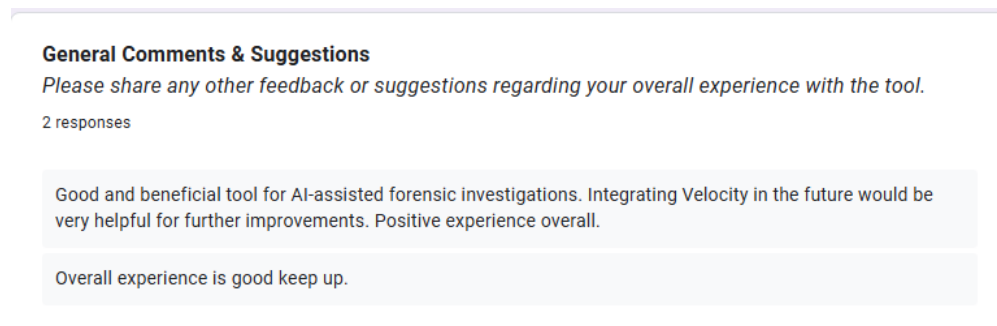


Figure 2.9: Practitioner General Comments and Suggestions



# Appendix C

## Results of Evaluation

This appendix provides supplementary screenshots or data relevant to the tool’s extended functionalities and interface designs, not included in the main text.

```
(dfllm_eval) D:\civyear\Research\Development\Evaluation_AutoDFBench>python Summary.py 66
(66, 10, 'ai_agents', 'windows_disk_path', {'shell', 'queued'}, 1, 1, 1)
['FT-SS', '02']
r_Base_Test_case02
02
[('unallocated', '0886'), ('deleted', '0885'), ('deleted', '0883'), ('deleted', '0881'), ('active', '0884'), ('active', '0882'), ('active', '0880'), ('active', '0852'), ('active', '0849'), ('active', '0850'), ('unallocated', '0854'), ('active', '0864'), ('active', '0866'), ('active', '0868'), ('deleted', '0865'), ('deleted', '0867'), ('deleted', '0902'), ('active', '0896'), ('deleted', '0901'), ('deleted', '0899'), ('deleted', '0897'), ('active', '0900'), ('active', '0898'), ('unallocated', '0918'), ('active', '0917'), ('deleted', '0915'), ('deleted', '0913')]
Matches with brackets: []
Matches with last word: [('0896', 'active')]
Matches with brackets: []
Matches with last word: [('0898', 'active')]
Matches with brackets: []
Matches with last word: [('0897', 'deleted')]
Matches with brackets: []
Matches with last word: [('0899', 'unallocated')]
Final Matches: {'deleted', '0897'}, ('active', '0898'), ('active', '0896'), ('unallocated', '0899')}
status: {'deleted', '0897'}, ('active', '0898'), ('active', '0896'), ('unallocated', '0899')}
true_positive_count: 3
total_tp_count: 3
total_fp_count: 1
total_fn_count: 27
precision: 0.75
recall: 0.1
f1: 0.17647058823529416
```

Figure 3.1: FT-SS-02 Test case results using auto df bench

```
r_Base_Test_case03
03
[('active', '0882'), ('active', '0884'), ('deleted', '0881'), ('deleted', '0883'), ('deleted', '0885'), ('unallocated', '0886'), ('active', '0880'), ('deleted', '0849'), ('unallocated', '0851'), ('active', '0852'), ('active', '0850'), ('active', '0848'), ('active', '0864'), ('active', '0866'), ('active', '0868'), ('deleted', '0865'), ('deleted', '0867'), ('deleted', '0869'), ('deleted', '0870')]
Matches with brackets: []
Matches with last word: [('0882', 'active')]
Matches with brackets: []
Matches with last word: [('0884', 'active')]
Matches with brackets: []
Matches with last word: [('0870', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0854', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0850', 'active')]
Matches with brackets: []
Matches with last word: [('0852', 'active')]
Matches with brackets: []
Matches with last word: [('0850', 'active')]
Matches with brackets: []
Matches with last word: [('0866', 'active')]
Matches with brackets: []
Matches with last word: [('0868', 'active')]
Final Matches: {'active', '0884'}, ('active', '0850'), ('active', '0882'), ('unallocated', '0870'), ('unallocated', '0854'), ('active', '0866'), ('active', '0852'), ('active', '0868')}
status: {'active', '0884'}, ('active', '0850'), ('active', '0882'), ('unallocated', '0870'), ('unallocated', '0854'), ('active', '0866'), ('active', '0852'), ('active', '0868')}
true_positive_count: 8
total_tp_count: 8
total_fp_count: 0
total_fn_count: 12
```

Figure 3.2: FT-SS-03 Test case results using auto df bench

```

(dfllm_eval) D:\4thYear\Research\Development\Evaluation_AutoDFBench>python Summary.py 68
(68, 10, 'ai_agents', 'windows_disk_path', {'shell'}, 'queued', 1, 1, 1)
['FT-SS', '04']
r_base_test_case04
04
[('active', '2944'), ('active', '2946'), ('active', '2948'), ('deleted', '2945'), ('deleted', '2947'), ('deleted', '2949')]
Matches with brackets: []
Matches with last word: [('0897', 'active')]
Matches with brackets: []
Matches with last word: [('0913', 'active')]
Matches with brackets: []
Matches with last word: [('0896', 'active')]
Matches with brackets: []
Matches with last word: [('0912', 'active')]
Matches with brackets: []
Matches with last word: [('0902', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0918', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0899', 'deleted')]
Matches with brackets: []
Matches with last word: [('0915', 'deleted')]
Matches with brackets: []
Matches with last word: [('0898', 'deleted')]
Matches with brackets: []
Matches with last word: [('0914', 'deleted')]
valuation_AutoDFBench > Summary.py

```

Figure 3.3: FT-SS-04 Test case results

```

r_base_test_caseRTL
RTL
[('deleted', '1513'), ('unallocated', '1537'), ('unallocated', '1535'), ('unallocated', '1533'), ('unallocated', '1531'), ('unallocated', '1529'), ('unallocated', '1497'), ('deleted', '1521'), ('deleted', '1519'), ('deleted', '1517'), ('deleted', '1515'), ('active', '1481'), ('deleted', '1511'), ('deleted', '1509'), ('active', '1501'), ('active', '1499'), ('active', '1477'), ('active', '1473'), ('active', '1493'), ('active', '1489'), ('active', '1485')]
Matches with brackets: []
Matches with last word: [('1481', 'active')]
Matches with brackets: []
Matches with last word: [('1485', 'active')]
Matches with brackets: []
Matches with last word: [('1489', 'active')]
Matches with brackets: []
Matches with last word: [('1493', 'active')]
Matches with brackets: []
Matches with last word: [('1497', 'unallocated')]
Matches with brackets: []
Matches with last word: [('1473', 'active')]
Matches with brackets: []
Matches with last word: [('1477', 'active')]
Final Matches: {'active', '1477'}, ('active', '1473'), ('active', '1485'), ('unallocated', '1497'), ('active', '1493'), ('active', '1489'), ('active', '1481')}
status: {'active', '1477'}, ('active', '1473'), ('active', '1485'), ('unallocated', '1497'), ('active', '1493'), ('active', '1489'), ('active', '1481')}
true_positive_count: 7
total_tp_count: 7
total_fp_count: 0
total_fn_count: 12
precision: 1.0
recall: 0.3684210526315789
f1: 0.5384615384615384

```

Figure 3.4: FT-07-RTL Test case results using auto df bench

```

ve', '0740'), ('active', '0672'), ('active', '0674'), ('active', '0676'), ('active', '0688'), ('active', '0690'), ('active', '0692'), ('active', '0686'), ('active', '0739'), ('deleted', '0741'), ('deleted', '0673'), ('deleted', '0675'), ('deleted', '0677'), ('deleted', '0689'), ('deleted', '0691'), ('deleted', '0693'), ('deleted', '0742'), ('unallocated', '0678'), ('unallocated', '0694'), ('unallocated', '0662')]
Matches with brackets: []
Matches with last word: [('0577', 'active')]
Matches with brackets: []
Matches with last word: [('0576', 'active')]
Matches with brackets: []
Matches with last word: [('0582', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0579', 'active')]
Matches with brackets: []
Matches with last word: [('0578', 'active')]
Matches with brackets: []
Matches with last word: [('0580', 'active')]
Matches with brackets: []
Matches with last word: [('0581', 'active')]
Final Matches: {'active', '0579'}, ('active', '0580'), ('active', '0578'), ('active', '0576'), ('active', '0577'), ('active', '0581'), ('unallocated', '0582')}
status: {'active', '0579'}, ('active', '0580'), ('active', '0578'), ('active', '0576'), ('active', '0577'), ('active', '0581'), ('unallocated', '0582')}
true_positive_count: 4
total_tp_count: 4
total_fp_count: 3
total_fn_count: 93
precision: 0.5714285714285714
recall: 0.041237113402061855
f1: 0.07692307692307693

```

Figure 3.5: FT-SS-09 results

```

ve, '0740'), ('active', '0672'), ('active', '0674'), ('active', '0676'), ('active', '0688'), ('active', '0690'), ('active', '0692'), ('active', '0698'), ('active', '0739'), ('deleted', '0741'), ('deleted', '0673'), ('deleted', '0675'), ('deleted', '0677'), ('deleted', '0689'), ('deleted', '0691'), ('deleted', '0693'), ('deleted', '0742'), ('unallocated', '0678'), ('unallocated', '0694'), ('unallocated', '0662')]
Matches with brackets: []
Matches with last word: [('0577', 'active')]
Matches with brackets: []
Matches with last word: [('0576', 'active')]
Matches with brackets: []
Matches with last word: [('0582', 'unallocated')]
Matches with brackets: []
Matches with last word: [('0579', 'active')]
Matches with brackets: []
Matches with last word: [('0578', 'active')]
Matches with brackets: []
Matches with last word: [('0580', 'active')]
Matches with brackets: []
Matches with last word: [('0581', 'active')]
Final Matches: {('active', '0579'), ('active', '0580'), ('active', '0578'), ('active', '0576'), ('active', '0577'), ('active', '0581'), ('unallocated', '0582')}
status: {('active', '0579'), ('active', '0580'), ('active', '0578'), ('active', '0576'), ('active', '0577'), ('active', '0581'), ('unallocated', '0582')}
true_positive_count: 4
total_tp_count: 4
total_fp_count: 3
total_fn_count: 93
precision: 0.5714285714285714
recall: 0.041237113402041855
f1: 0.07692307692307693

```

Figure 3.6: FT-SS-10 results

# Appendix D

## Github Repository

The complete source code for the AI-driven digital forensics framework is available at the following GitHub repository:

`Github Repository`

This repository contains the implementation of the multi-agent architecture, the integration with The Sleuth Kit (TSK), Retrieval-Augmented Generation (RAG) techniques, ReAct prompting methods, the evaluation scripts used for AutoDFBench benchmarking, and supplementary materials related to this research.