INTELLIGENT AGENTS: DEVELOPMENT TEAM PROJECT

GROUP E: Murthy Kanuri, Jaco Espag, Finlay Kirwan, Geng Jittipattanakulchai

Module: Intelligent Agents – IA_PCOM7E July 2025 A

Tutor: Dr Samuel Danso

8th September 2025

Intelligent agents | Digital Forensics

INTRODUCTION

This report outlines the design proposal for an intelligent agent system developed within the domain of digital forensics. The agent is intended to automate the process of identifying and retrieving specified file types from a file system, processing the information, and securely archiving them for subsequent analysis. The following proposal outlines the system requirements, design decisions, and methodological approaches that will guide the development of a reliable and scalable solution.

SYSTEM REQUIREMENTS

The agents will be developed in Python 3.11 or a later version to ensure cross-platform compatibility and to benefit from modern library support. The use of standard Python libraries for file system traversal and metadata extraction will further contribute to a consistent experience across platforms.

The use of psutil will provide environment awareness by detecting mounted and temporary volumes, supporting system safety through the exclusion of critical directories on the supported platforms (Rodola, 2025). For the purposes of file identification, we will employ the python-magic library to perform MIME-type analysis, as opposed to relying on less reliable filename extensions. The filetype library will be incorporated as a lightweight alternative to ensure compatibility when needed (Hupp, 2022; Tom, 2022).

Even though the agent will be aware of common critical directories, to further limit the potential for system instability, the agent will only store metadata. A secondary agent will learn from manual archiving behaviour to automate future archival tasks. This learning

component will use scikit-learn for supervised modelling and joblib for model persistence and optimisation (Joblib Developers, 2025; scikit-learn Developers, 2025). While the use of Python's concurrency capabilities is optional, they may support future scalability. File metadata will be stored in an SQLite database, chosen for its ease of use, portability, and query support (SQLite Consortium, 2025).

Non-functional requirements include a focus on the ethical use and storage of data, cross-platform consistency, minimal configuration, and a modular, maintainable codebase suitable for forensic use.

DESIGN DECISIONS

The main objective of the agent is to find and retrieve specific file types based on pre-defined rules. To achieve this, the agent architecture is based on a hybrid reactive pipeline (Mohan., 2025) that provides adaptive but predictable behaviour through a process of discover-identify-decide-act-audit. Runs will be initiated using a command-line interface for reproducibility and autonomy.

The agent scans for files on a system, then performs type identification using content-based methods such as python-magic, with file extension checks as a fallback. Once the correct types are identified, the metadata will be processed and saved to a database for analysis. As no single Python library consistently exposes metadata across all file types, the parsers within the Python script will need to be modified depending on the end use of the agent. Security shall be maintained by restricting operations to read-only sources and controlled destinations, ensuring the agent preserves evidential integrity while remaining adaptable to diverse forensic scenarios. See figure 1 for the high-level

overview, and figures 2, 3 and 4 for the proposed class diagram and component interactions.

Figure 1: High-Level Logic Flow Diagram

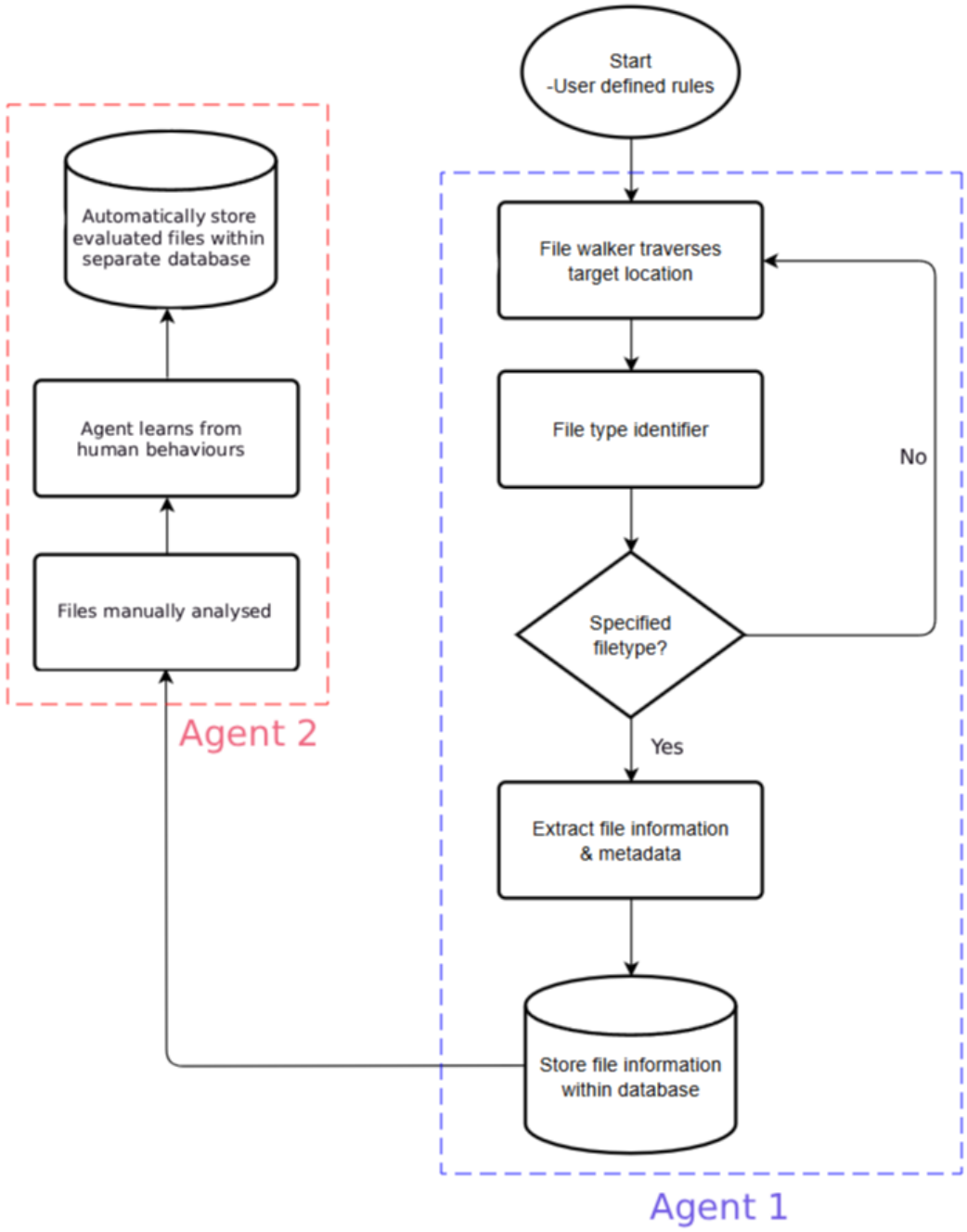


Figure 2: Class diagram

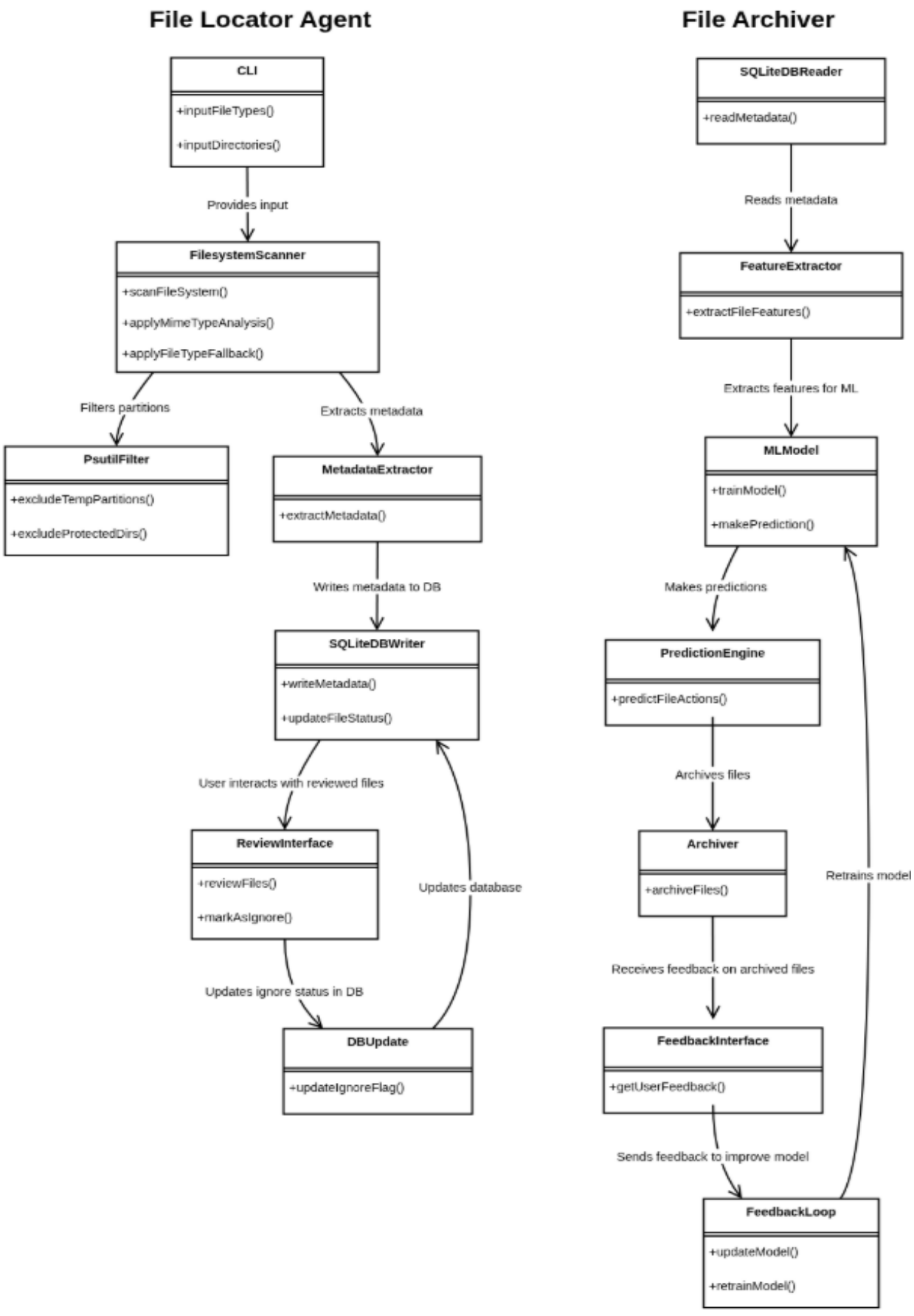


Figure 3: Metadata Extraction

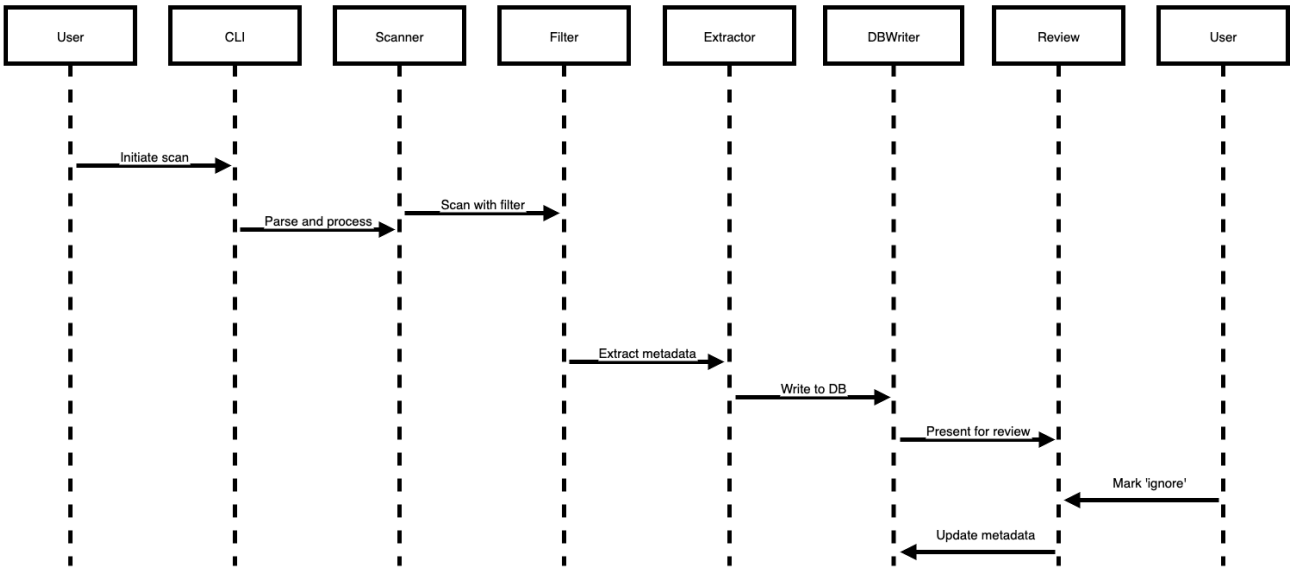
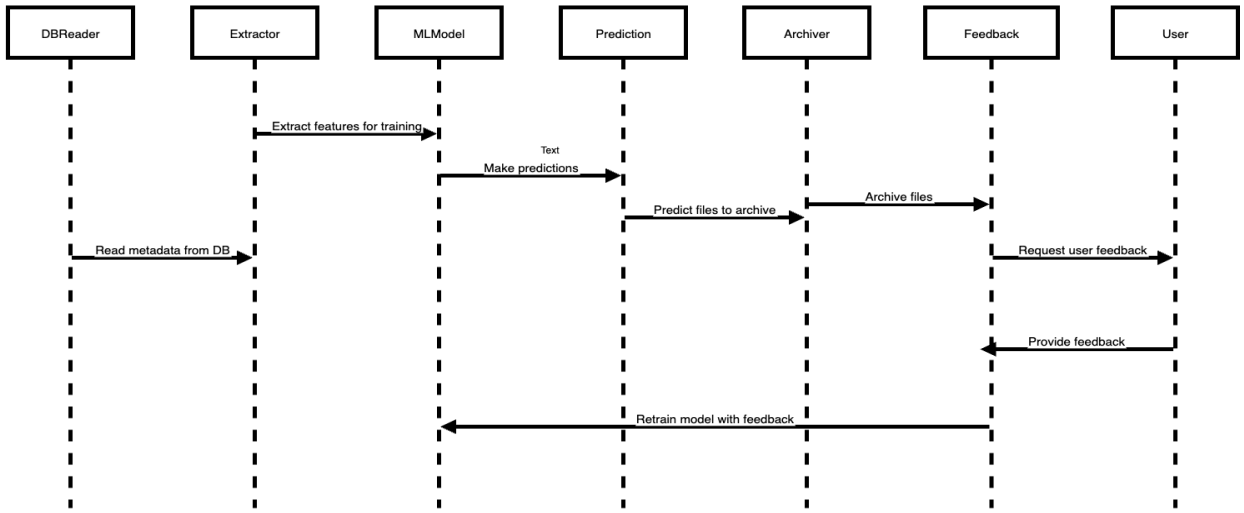


Figure 4 : File Archiving



METHODOLOGY

We adopted an Agile development methodology to support the iterative nature of building intelligent agents for digital forensics. During development, challenges such as refining type-identification methods, adapting to cross-platform differences, or the need for alternative Python libraries may arise unexpectedly. The workload will be structured

into fixed-length sprints, with each sprint focused on one component of the pipeline. For example, the discovery and identification phase will be fully implemented and tested before advancing to the processing or audit stages.

The development of the agent will follow the rules of Agent-Oriented Programming (AOP), keeping code organised and focused (Shoham, 1993). The agent will think in terms of its own beliefs, goals and actions. For example, "I know where to look", "I need to find these file types" and "I'm going to extract its metadata".

Finally, since we are building each part as a separate, reusable module, we'll use object-oriented principles to make the code easier to maintain and extend in the future (Parnas, 1972).

CRITICAL EVALUATION

The proposed intelligent agent system presents a design based on our understanding of the core needs in digital forensics. A major strength of the design lies in its modular architecture, which not only supports maintainability and scalability but also allows the individual components for scanning and identification, and archiving to evolve independently. Using well-supported open-source libraries helps keep the system flexible across different platforms while also reducing the amount of custom development required.

Metadata, while less intrusive than full file contents, can still include personally identifiable information. In line with data minimisation principles (ICO, 2024), only essential metadata will be retained. This safeguards both user privacy and the forensic integrity of the process.

That said, several challenges need to be addressed. The current design relies heavily on MIME-type analysis to detect file types. While this method is generally more reliable than checking file extensions, it is not foolproof and may behave unpredictably across different file formats (Dubettier et al., 2023).

Another concern lies in the use of supervised learning for automating the archiving process. This introduces a dependency on well-labelled training data; if the model isn't validated regularly, there's a risk of misclassification (Lopez et al., 2023).

Perhaps the most critical risk is system stability. Since the tool is guided by user input and may be run with elevated permissions, there's a possibility that it will archive files from sensitive or critical areas by mistake. Without strict controls in place, this could lead to data loss. Safeguards, such as explicit exclusions and dry-run modes, should be considered to avoid unintended consequences.

CONCLUSIONS AND RECOMMENDATIONS

In this work, we examined how an intelligent agent can be designed to support digital forensic processes. By utilising a hybrid reactive model, we separate the core use case of locating files from the archiving part, ensuring we do not run the risk of potentially impairing the system while maintaining flexibility to respond to different scenarios without sacrificing consistency.

During the implementation, we will test the agent across multiple platforms to ensure consistent performance, particularly when handling file metadata and uncommon file types. In designing the agent, consideration was given to the merits of content-based identification; however, it should not be assumed flawless, and the proposed fallback strategies must be refined in line with our agile development approach.

Finally, due to the proposed use of machine learning to aid in the automated archiving of files, it is important to monitor performance over time, ensuring the agents remain dependable in performing their activities.

REFERENCES

Dubettier, A., et al. (2023) 'File type identification tools for digital investigations', *Forensic Science International: Digital Investigation*, 46, p. 301574. doi:10.1016/j.fsidi.2023.301574.

Hupp, A. (2022) *python-magic*. Available at: https://github.com/ahupp/python-magic (Accessed: 2 September 2025).

ICO (2024) *Data minimisation*. ICO. Available at: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/data-minimisation/ (Accessed: 3 September 2025).

Joblib Developers (2025) *Joblib documentation*. Available at: https://joblib.readthedocs.io/ (Accessed: 2 September 2025).

Lopez, E., et al. (2023) 'The importance of choosing a proper validation strategy in predictive models. A tutorial with real examples', *Analytica Chimica Acta*, 1275, p. 341532. doi:10.1016/j.aca.2023.341532.

Mohan, R.N.V.J., Raju, B.H.V.S.R.K., Sekhar, V.C. and Prasad, T.V.K.P. (2025) *Algorithms in Advanced Artificial Intelligence*. 1st edn. CRC Press, pp. 738–743. doi:10.1201/9781003641537 (Accessed: 1 September 2025).

Parnas, D.L. (1972) 'On the criteria to be used in decomposing systems into modules', *Communications of the ACM*, 15(12), pp. 1053–1058. doi:10.1145/361598.361623 (Accessed: 4 September 2025).

Rodola, G. (2025) *psutil*. Available at: https://github.com/giampaolo/psutil (Accessed: 2 September 2025).

scikit-learn Developers (2025) *scikit-learn*. Available at: https://github.com/scikit-learn/scikit-learn (Accessed: 2 September 2025).

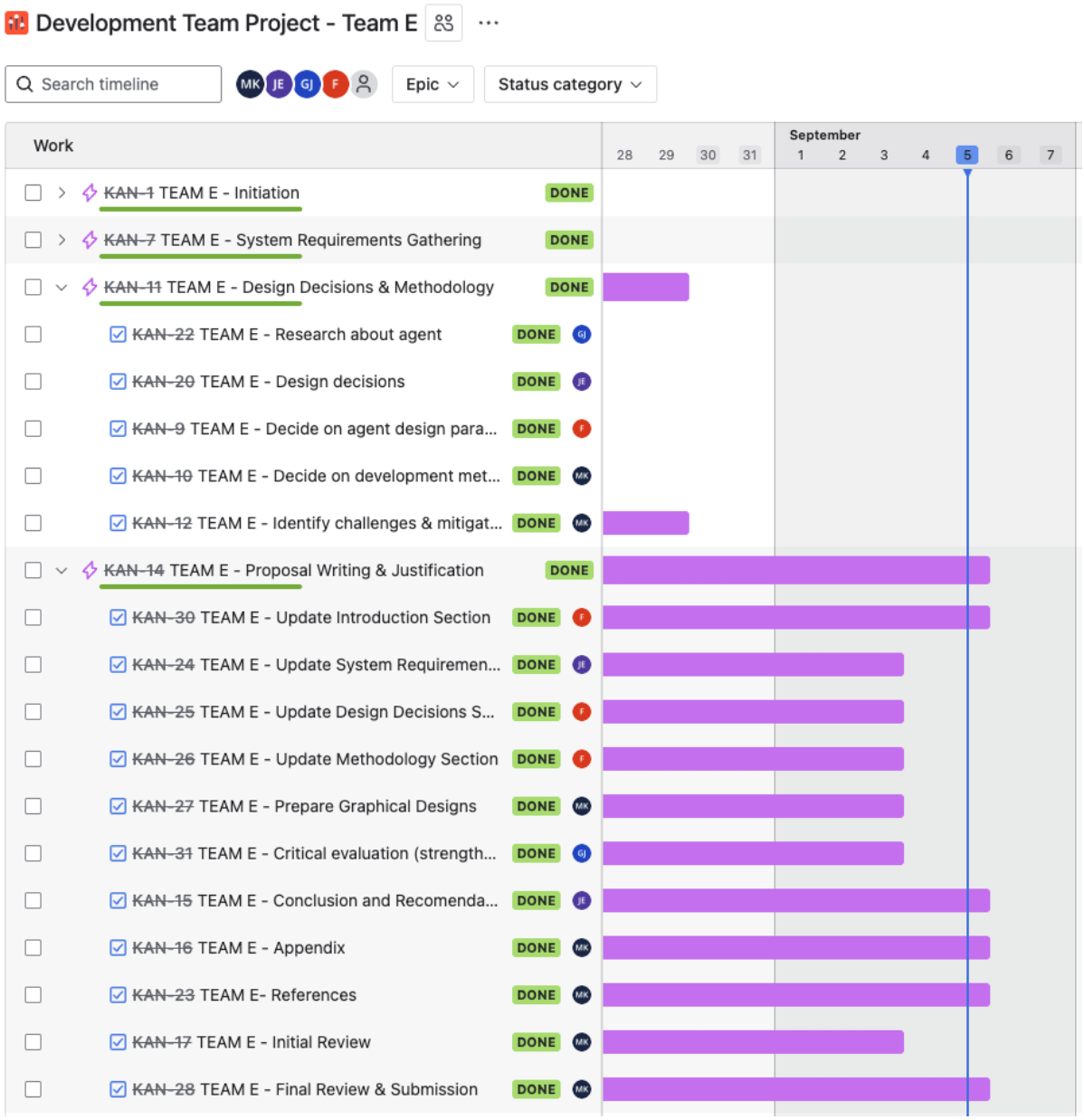
Shoham, Y. (1993) 'Agent-oriented programming', *Artificial Intelligence*, 60(1), pp. 51–92. doi:10.1016/0004-3702(93)90034-9 (Accessed: 4 September 2025).

SQLite Consortium (2025) *SQLite*. Available at: https://www.sqlite.org (Accessed: 2 September 2025).

Tom (2022) *filetype*. Available at: https://github.com/h2non/filetype.py (Accessed: 2 September 2025).

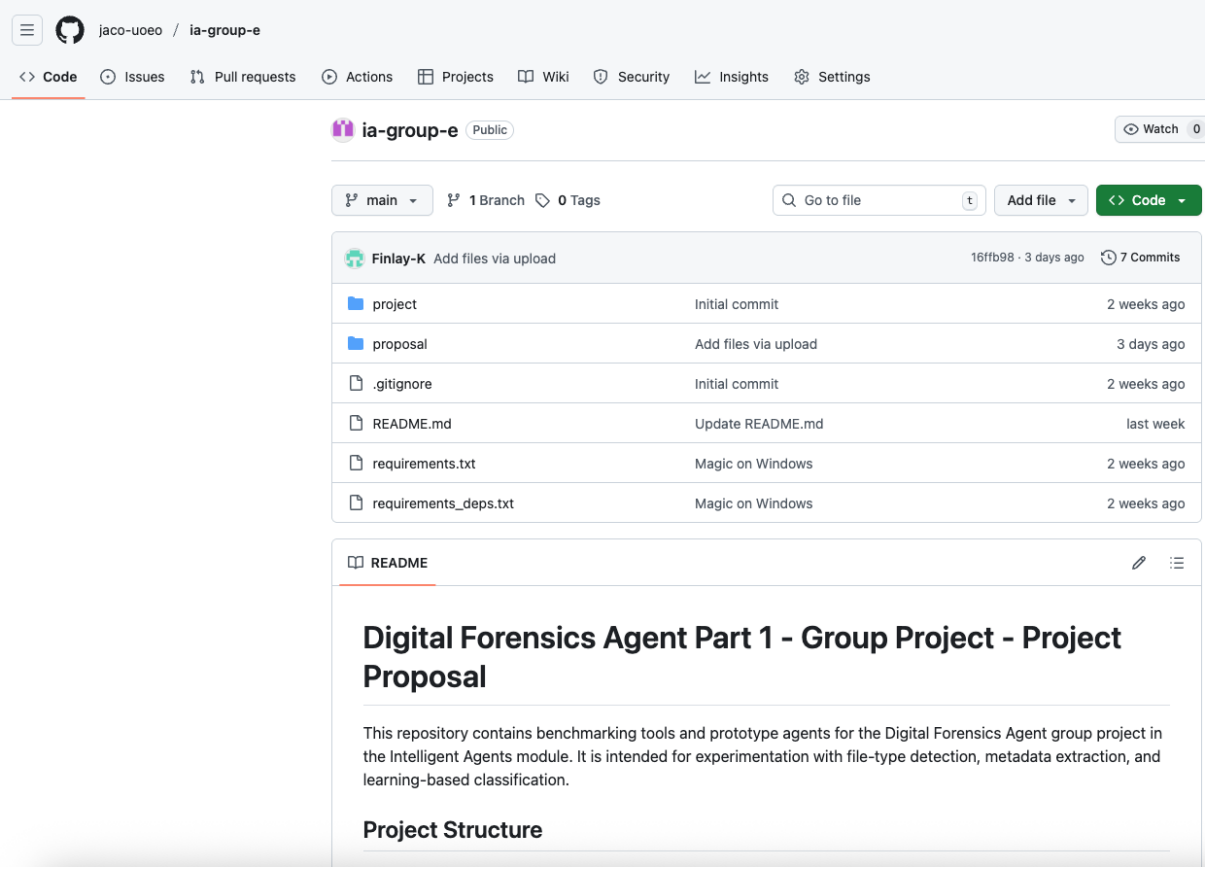
APPENDIX

JIRA BOARD VIEW



GITHUB VIEW

https://github.com/jaco-uoeo/ia-group-e/tree/eaa99d87f9efe5454492341cdaedbf727ad95af5



Google Drive