

# Realistic Log Generation through Large Language Models: Bridging Data Gaps in Log Analysis

Athira Gopal<sup>1</sup> and Supriya Bajpai<sup>2\*</sup>

<sup>1</sup>Computational and Data Science Department, Indian Institute of Science (IISC),  
Bengaluru, Karnataka, 560012, India.

<sup>2</sup>IIT Bombay & Monash University, IIT Bombay, Mumbai, Maharashtra, 400076,  
India.

\*Corresponding author(s). E-mail(s): [supriyamishra39@gmail.com](mailto:supriyamishra39@gmail.com);  
Contributing authors: [athira.anudarshan@gmail.com](mailto:athira.anudarshan@gmail.com);

## Abstract

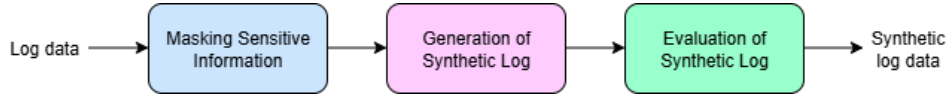
System logs are very useful for monitoring, debugging and diagnosing failures of large systems but they tend to be sensitive and thus not easy to share and make use of in a research. We propose a solution to this problem: to produce synthetic logs information of sufficiently good quality such that the structure and the statistical distribution of the real logs and privacy information are preserved. The strategy has three steps. The former part is masking the sensitive material on the logs by using a multi-pass algorithm of anonymization that generates a dictionary of sensitive words and substitutes them with generic names. Second, using the anonymized logs synthetic logs are generated using large language models (LLMs) and to ensure continuity between chunks summaries of previously generated synthetic log chunks are used and reflective verification loops also. Lastly, the produced logs are fed through a multi-agent debating model which is automated and functions using an LLM pipeline to achieve the structural and contextual correctness. With the help of this model, organizations can generate viable, interchangeable logs without revealing information that is confidential that can be used to aid in the completion of tasks such as log enquiry, failure enquiry, and root-cause research.

**Keywords:** Generative AI, AI agent, Synthetic log generation, Large Language Models (LLMs)

## 1 Introduction

The large content in the system records concerning the organization is highly sensitive e.g. infrastructure, behavior and activities of the system by users and therefore, this should not be shared with people. To overcome this shortcoming, this paper targets to make a system for synthetically generating system logs, which has the same structure and characteristics of the private logs within organizations.

A synthetic log data is applied in most of the real-life applications in the field of IT work and cybersecurity, as well as in research. It gives an opportunity to analyze the behavior and the



**Fig. 1:** Overview of 3 Stage Synthetic Log Generation

performance of a system without compromising the internal confidential information. Synthetic logs are most often used in root cause analysis to help engineers in the investigation of failures without even having to look at the actual production logs. They are also handy in designing and testing of IT systems, detection and monitoring of anomalies where the actual data of the logs are not available to share. Synthetic logs are also used to benchmark and train machine learning models and to execute large-scale experiments because they not only give realistic data but also do not violate privacy.

Among such key issues that can be faced when constructing synthetic logs, major one is the excessive volume of system logs data. The system logs of most cases are larger than the context window of a large language model and therefore the model can only fit a portion of the data at a particular time. Regardless of the sophistication of our models, such as GPT-4o, the quantity of tokens that can be generated or processed at a single run to create the synthetic logs, is limited.

The other critical issue is that of evaluating the generated synthetic logs. Because synthetic logs contain non-natural language data, manual evaluation is difficult and automated procedures are only possible evaluation methods. The quality and accuracy of synthetic logs can be assessed with the help LLM based evaluation methods.

The combination of large language models and anonymization principles and a special evaluation platform can produce quality synthetic data and this method protects confidential data from being leaked. The given method is not confined to the system logs as it can be applied in the other fields where the structured information has to be extracted and privacy has to be preserved.

This paper has been able to create the synthetic log without the leakage of any sensitive information. The system logs are anonymized by replacing the confidential fields with generalized placeholders. Large language models are then fed with the synthesized log data in the form of chunks. The generated logs are extremely huge and to check structural validity of them, we employ a reflective step in checking logs. We check continuity of the generated log chunk by the use of summary of the preceding log chunks. Finally, the synthetic logs are also tested with the aid of multi-agent architecture that verifies the structural correctness and continuity of generated synthetic logs.

The main contribution of this paper are follows,

- A method for anonymizing sensitive content in system log data.
- An approach for generating synthetic log data from anonymized logs.
- A framework for evaluating the correctness and quality of the generated logs.

## 2 Related Work

Synthetic logs can be used in a wide variety of tasks, including question answering, anomaly detection, and root cause analysis. The framework in [1] offers effective question answering system which can be applied on complicated IT logs comprising domain terminology, error codes and file names. The LogHub dataset [2, 3], offers research log data on various tasks.

There are also some approaches that produce synthetically generated logs through the traditional or large language models. AnomalyGen [4, 5] creates semantic log sequences by combining static program analysis and Chain of Thought prompting approach using LLMs. AnomalyGen focuses to generate synthetic logs for anomaly detection. But our method is more flexible and it can generate the synthetic log for any task.

LogGenST method [6–8] uses three connected LLMs to generate realistic logs. It has 3 components, Gnerator, Discriminator and Prompt Engineer. The Generator creates the logs, Discriminator checks their quality, and Prompt Engineer updates the instructions based on this feedback. This process helps ensure that system faults are represented correctly. However, this method does not use raw, unprocessed logs. It also relies only on a single discriminator LLM for evaluation. In contrast, our approach includes a reflective verification step during the generation of each log chunk. Additionally, every generated chunk is evaluated independently using a multi-agent debating system.

[9] describes a non-LLM method for generating synthetic log files. It uses Generative Adversarial Networks (GANs) to make the logs look realistic. This approach requires GANs to be trained on processed log data.

Our framework gives more importance to privacy than existing methods. At the same time, it preserves the structure and flow of real log data. LogGenST uses an adversarial LLM to generate fault scenarios without using real logs. Its main focus is flexibility, not privacy protection. AnomalyGen, relies on static program analysis and access to source code. It generates semantically rich log sequences, mainly for anomaly detection. It does not directly address privacy concerns. Although all three approaches use LLMs to generate realistic logs, our framework offers an advantage. It enables the creation of shareable, privacy-preserving logs from real world data.

## 3 Methodology

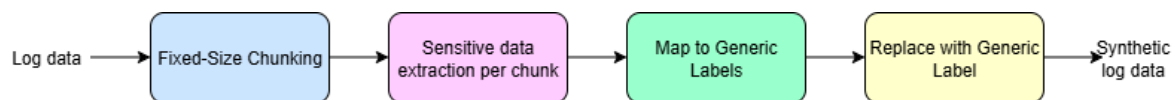
### 3.1 Problem Statement

System logs play a valuable role IT systems since it contain sensitive information about organizations and users, they are difficult to share. Synthetic logs should be made in such a way that preserve the structure and the patterns of the original data and while eliminating the privacy issues. The most important tasks in generating the synthetic logs are concealing sensitive information, producing high quality synthetic logs and ensuring the continuity and structural correctness.

### 3.2 Masking sensitive information

Examples of sensitive information that is usually present in system log data are system architecture and system user activity of an organization. Because the system logs are large in size, it is not possible to mask all the sensitive contents in one step.

To handle large log files, the log data is split into fixed size chunks. A well written prompt is used to detect sensitive data in each log chunk. All the collected sensitive data is grouped based on its categories and a generic label is assigned for each categories. The collection of sensitive content is done by a local LLM. De-duplication of sensitive content is also performed. By this process a dictionary is generated which connect sensitive content to its category name. With this dictionary, sensitive content in the log file is programatically replaced by its category name.



**Fig. 2:** Diagram of Masking Mensitive Content from the Log File

### 3.3 Generation of synthetic log

The second step is the synthesis of log information of the anonymized (masked) logs after masking the sensitive information. Proprietary models (e.g. GPT 4o) can exist because of the substitution of the sensitive fields. The masked log is divided into model context-window limited blocks.

It is created in batches and each batch is inputted into the LLM and few-shot prompt is provided which explicitly requests the model not to do the word-to-word recreation, but compels them to create a log structure. The first chunk contains the masked chunk and examples in the few-shot prompt are used to convince the generator to learn the format of the target.

It is challenging to keep continuity among pieces. To ensure the integrity and retain the immediate succinct we retain a brief outline of each of the generated chunks. The masked form of the last chunk  $i$ , the most recently generated chunk  $i$ , and summaries of the last  $n$  chunks generated are given by the fact that the prompt of generation of chunk  $i$  is more than 1. This assists in ensuring that it offers adequate background that does not overstrain in the provision of the input of the model.

The quality of the generated logs may begin to decrease with the increase of the number of chunks. To cope with this we resort to reflective verification loop. Following the creation of chunk, an independent verification prompt will be used to determine whether the chunk is structurally sound and whether even it will fit within the already created content. The checking person passes or fails and also provides brief explanation. As the chunk passes then it is retained and the generation is resumed. When the verifier and original masked chunk fail to be recognized, the previous generated chunk and corresponding summaries are sent through the generator which can then generate an improved one.

### 3.4 Evaluating the generated synthetic data

Synthetic log data generated should be evaluated. Since human evaluation is expensive and practically difficult, a better option is to use LLMs for synthetic log evaluation.

This algorithm involves a few-shot prompt to match each chunk of the log generated under this technique with the original one. In order to give it a context, the chunk generated immediately before is also evaluated alongside it in form of summaries of the preceding ten chunks. The evaluation is conducted within a multi-agent deliberation system where one of the agents identifies the affirmative aspect of the formulated chunk, the other agent emphasizes the demerit aspect and a judge is brought to determine whether the chunk is connected to the standards of quality demanded.

This synthetic log evaluation approach uses a multi-agent debating method. To check the continuity of the generated log chunk, the log chunk is evaluated with summaries of the previous ten chunks and the previous chunk itself. To check the structural correctness, each synthetically generated log chunk is compared with the original log chunk. In the debating system, one agent lists the strengths of the generated chunk ( both structural and continuity) , another lists its weaknesses, and the judge decides whether the chunk has necessary qualities. The judge can ask questions to the agents, this makes the evaluation interactive.

## 4 Experimental Details

### 4.1 Preprocessing setup

To hide the sensitive content, the log file is divided into fixed size chunks of 200 lines. Sensitive information in each log chunk is collected using the Mistral-7B model with the help of a structured few-shot prompt. The collected sensitive information is stored in a text file as a dictionary. In the dictionary each category name corresponds to the sensitive entities it represents.

After all chunks are processed, a Python script collect sensitive contents from all the log chunks and removes duplicate entries. This gives a complete dictionary for the log file that maps each sensitive data to its category name. Finally, another Python script replaces all sensitive information in the log file with the corresponding category name.

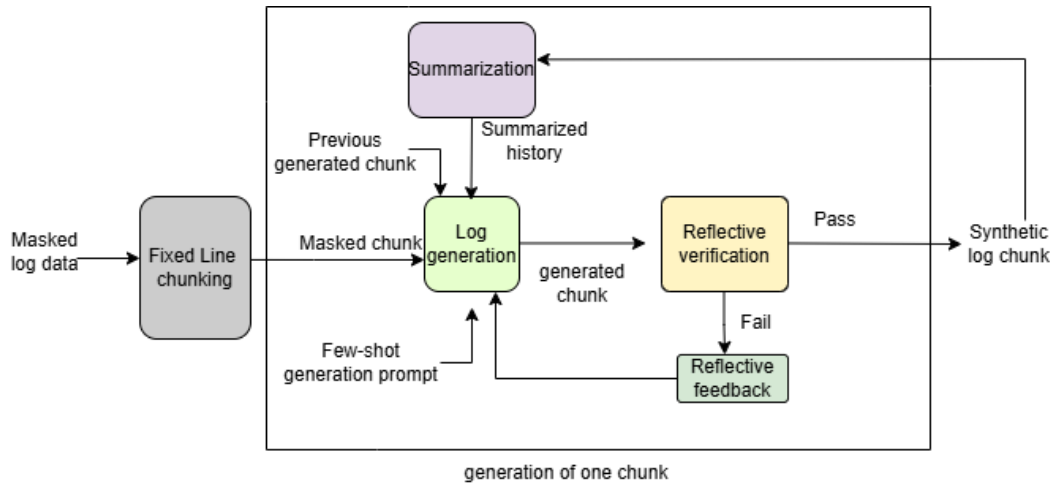


Fig. 3: Workflow of Synthetic Log Generation from Anonymized Logs

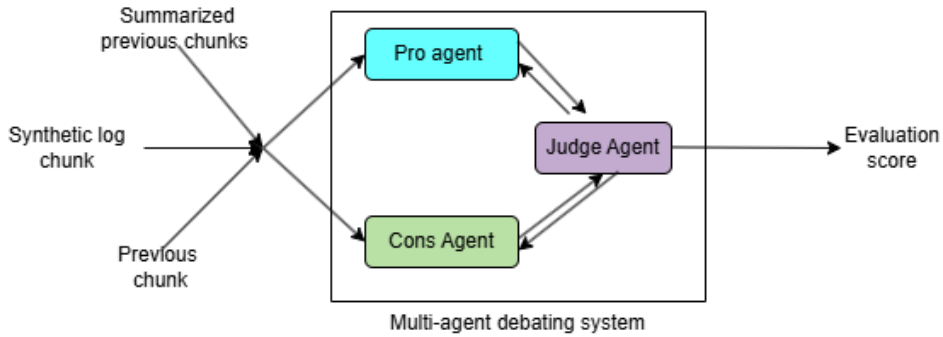


Fig. 4: Multi-Agent Debating System for Evaluating Synthetic Logs

## 4.2 Synthetic log generation

Synthetic logs are generated from the anonymized log file using GPT-4o model. The anonymized log data is split into 200 line chunks and generated one chunk at a time. To make the generated log continuous, the model is given examples, short summaries of the previous ten blocks, and the most recently generated chunk.

After generation of log chunk, each generated log chunk is evaluated to confirm its structural correctness and continuity. If any issues are found in the generated log chunk, feedback is given to the log generator and the chunk is regenerated until it meets the required quality.

## 4.3 Evaluation framework

To evaluate the correctness of generated log chunk, multi agent debating system is used with GPT-4o as the base model.

## 4.4 Implementation Details

Sensitive information was anonymized using Mistral-7B model and to generate and evaluate the synthetic logs GPT-4o was used. For log generation temperature of 0.1 and verification and evaluation temperature of 0.01 are used. The number of tokens was kept within the model’s context window.

Experiments were done on real system logs called Warrior logs. The Warrior logs are semi-structured. They store information related to timestamps, severity information, and event messages.

Mistral-7B model was run on a NVIDIA A40 GPU (24 GB VRAM). There was no finetuning, all models are used via inference.

## 5 Limitations

The synthetic log generation method works well, but it has some limitations. One important issue is hallucinations, the data generated by the framework seem to be correct, but may not be correct. Reflective verification step and multi-agent evaluation system catch many errors, but they cannot identify all the errors.

The method relies on GPT-4o. This make the log generation easier but limit the access and reproducibility of the method. Using open-source could help in the future.

Generating logs in chunks and checking their quality repeatedly increases computation cost. This can slow down the system and make the scaling also harder. This could be improved by increasing chunk sizes and reducing evaluations.

Overall, these limitations show the trade-offs between quality, and efficiency, and point to ways the framework can be improved.

## 6 Conclusion

This paper presents a new method to create and evaluate synthetic log data from already existing log files that contain sensitive information. The method combines anonymization process, use of large language models, and step-by-step evaluation using multi agent debating system. This framework ensures both structural and continuity correctness of the generated log data. Logs are generated in chunks, checked with reflective verification, and reviewed by a multi-agent system to ensure quality and accuracy. This allows organizations to safely use synthetic logs for analysis, anomaly detection, and benchmarking without revealing sensitive information. Overall, the approach shows that modern language models can help produce high-quality, privacy-safe synthetic logs.

## Declarations

- The authors received no specific funding for this study.
- The authors declare that they have no conflicts of interest to report regarding the present study.
- No Human subject or animals are involved in the research.
- All authors have mutually consented to participate.
- All the authors have consented the Journal to publish this paper.
- Authors declare that all the data being used in the design and production cum layout of the manuscript is declared in the manuscript.

## References

- [1] Bajpai, S., Kumar, A.G.C.H.N.: Hg-insightlog: Context prioritization and reduction for question answering with non-natural language construct log data

- [2] Zhu, J., He, S., He, P., Liu, J., Lyu, M.R.: Loghub: A large collection of system log datasets for ai-driven log analytics. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), pp. 355–366 (2023). IEEE
- [3] Zhu, J., He, P., Liu, J., He, S., Lyu, M.R.: Tools and benchmarks for automated log parsing. In: Proceedings of the 41st International Conference on Software Engineering (ICSE), pp. 121–130 (2019). IEEE
- [4] Li, X., Huo, Y., Mao, C., Shan, S., Su, Y., Li, D., Zheng, Z.: Anomalygen: An automated semantic log sequence generation framework with llm for anomaly detection. arXiv preprint arXiv:2504.12250 (2025)
- [5] Fariha, A., Gharavian, V., Makrehchi, M., Rahnamayan, S., *et al.*: Log anomaly detection by leveraging llm-based parsing and embedding with attention mechanism. In: 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (2024). <https://doi.org/10.1109/CCECE59415.2024.10667308>
- [6] Partovian, S., Flammini, F., Bucaioni, A., Thornadtsson, J.: Loggenst: A framework for synthetic log generation using llms for smart-troubleshooting. In: Topical Area: Software, System and Service Engineering (S3E) of the FedCSIS Conference on Computer Science and Intelligence Systems, pp. 64–83 (2024). Springer
- [7] Zhang, Z., Li, S., Zhang, L., Ye, J., Hu, C., Yan, L.: Llm-lade: Large language model-based log anomaly detection with explanation. Knowledge-Based Systems **326**, 114064 (2025)
- [8] Nadas, M., Diosan, L., Tomescu, A.: Synthetic data generation using large language models. arXiv preprint arXiv:2503.14023 (2025)
- [9] Partovian, S., Flammini, F., Bucaioni, A.: Leveraging gans to generate synthetic log files for smart-troubleshooting in industry 4.0. In: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 475–482 (2024). <https://doi.org/10.1109/SEAA64295.2024.00079>

## 7 Appendix

### 7.1 Prompts

These are the prompts used for the experiments.

**Synthetic Log Generation Prompt="**  
You are given a chunk of system log data that has already been anonymized. Your task is to generate a synthetic version of the given log chunk that:

- Preserves the overall structure, style, and format of the original log data.
- Introduces realistic but fictitious content, ensuring no line is copied verbatim from the original.
- Maintains logical continuity with previously generated chunks.

You will be provided with:

- The anonymized log chunk to transform.
- The most recently generated chunk (for direct continuity).
- Summaries of the 10 preceding generated chunks (for broader context).

**Guidelines:**

- Keep timestamps, event sequences, and formatting consistent with typical log file patterns.
- Ensure the generated chunk smoothly connects with prior logs (e.g., user sessions, request flows, error sequences).
- Do not fabricate unrealistic or inconsistent entries (e.g., impossible timestamps, contradictory user actions).
- Avoid copying lines directly from the original input — generate new but structurally similar log entries.
- If the original chunk contains repetitive structures (e.g., multiple login attempts), preserve the pattern, but vary the content appropriately.

**Example:**

**Original log chunk (anonymized):**

```
2023-08-14 10:45:12 INFO User_A logged in from 10.0.0.5 on host app-server-01
2023-08-14 10:45:15 ERROR Timeout connecting to db-server-03
2023-08-14 10:45:20 INFO User_A retried request to /api/data
```

**Synthetic log chunk (generated):**

```
2023-08-14 11:02:08 INFO User_B authenticated from 192.168.2.45 on host web-node-02
2023-08-14 11:02:10 ERROR Failed to establish connection with cache-node-01
2023-08-14 11:02:14 INFO User_B attempted request to /service/query
```

**Instructions:**

Now, generate the synthetic version of the following log chunk, ensuring it aligns structurally with the input while maintaining continuity with the provided summaries and preceding chunk.

**Fig. 5:** Prompt for synthetic log generation

**Masking prompt=**

You are given a chunk of system log data that may contain sensitive information.  
Your task is to identify and extract all sensitive entities from the chunk and present them in a structured dictionary-style format.

Sensitive information includes (but is not limited to):

- IP addresses (e.g., 192.168.0.1, 10.0.0.5)
- Username (e.g., jdoe, admin\_user, alice123)
- User email addresses (e.g., john.doe@example.com,alice@mail.org)
- Physical addresses (e.g., 123 Main St, Springfield, IL, 742 Evergreen Terrace, Springfield)
- System or hostnames (e.g., webserver01, db-prod-03, cache01.internal)
- API keys, passwords, or tokens (e.g., AKIAIOSFODNNEXAMPLE, p@ssw0rd123)
- Other personally identifiable information (PII) or organization-specific identifiers

Examples:

**Example 1:**  
Log chunk: User jdoe logged in from IP 192.168.0.1 on host webserver01

**Output:**  
Username: jdoe  
IP address: 192.168.0.1  
System name: webserver01

**Example 2:**  
Log chunk: Email alice@mail.org accessed database db-prod-03 from IP 10.0.0.5.

**Output:**  
Email: alice@mail.org  
System name: db-prod-03  
IP address: 10.0.0.5

**Example 3:**  
Log chunk: Payment received from John Doe at 123 Main St, Springfield, IL using API key AKIAIOSFODNNEXAMPLE.

**Output:**  
Username: John Doe  
Physical address: 123 Main St, Springfield, IL  
API key: AKIAIOSFODNNEXAMPLE

**Instructions for New Log Chunks**

- Carefully scan the given log chunk.
- Extract all sensitive entities (usernames, emails, IPs, system names, physical addresses, API keys, tokens, etc.).
- Present them in the same structured format as in the examples above.
- Do not include non-sensitive log text.
- Always use clear labels (e.g., Username, IP address, System name).
- This makes it strict, reproducible, and easy for the LLM to stay consistent.

**Fig. 6:** Prompt for masking sensitive content from log data

**Reflection prompt= You are given:A newly generated synthetic log chunk**

- *The original anonymized log chunk from which it was generated.*
- *The immediately preceding generated chunk.*
- *Summaries of the 10 most recently generated chunks (for broader context).*

**Your task is to evaluate the generated log chunk based on the following criteria:**

**Evaluation Criteria**

**Structural Correctness:**

- *Does the generated chunk follow the same log format as the original?*
- *Are timestamps, log levels, and field structures consistent and plausible?*
- *Are repeated patterns, sequences, or entry types preserved correctly?*
- *No lines should be identical copies from the original log.*

**Continuity:**

- *Does the generated chunk connect logically with the previous chunk?*
- *Are the events and sequences in the right order and context across chunks?*
- *Are ongoing sessions, transactions, or error flows consistent with previous chunks?*

**Instructions for Response:**

- *Check both structural correctness and continuity separately.*
- *Give a pass or fail rating for each category.*
- *If either category fails, give a clear reason showing why the generated chunk doesn't meet the standards.*
- *Suggest corrections or improvements to help regenerate the chunk correctly.*

**Output your evaluation in the following format:**

**Structural Correctness: PASS/FAIL**

**Reasoning: [Explain if FAIL]**

**Continuity: PASS/FAIL**

**Reasoning: [Explain if FAIL]**

**Overall Assessment: PASS/FAIL**

**Recommendations: [If FAIL, provide suggestions for improvement]**

**Example Output**

**Structural Correctness: PASS**

**Reasoning: The chunk keeps the correct log format, including timestamps, log levels, and field structure, and does not repeat any lines from the original.**

**Continuity: FAIL**

**Reasoning: The events in this chunk skip over a user session from the previous chunk, disrupting the sequence.**

**Overall Assessment: FAIL**

**Recommendations: Include the missing user session events and ensure the timestamps continue in the correct order from the previous chunk.**

**Fig. 7:** Diagram for synthetic log evaluation

**Pros Agent Prompt(Strengths Evaluation)=**

**Context:** You are the Pros Agent in a multi-agent evaluation framework. You are tasked with identifying all the strengths of a generated synthetic log chunk. Your evaluation should check this.

**Structural fidelity:** the log format is correct, fields are consistent, timestamps are logical, and the overall style is preserved.

**Continuity:** logical flow and coherence with the previous chunks and prior log summaries.

**Input Provided:**

- Generated log chunk
- Corresponding original anonymized chunk
- The previously generated chunk
- Summaries of the 10 previously generated chunks.

**Instructions:**

- Identify all parts of the generated chunk that match the expected log format.
- Highlight the patterns, sequences, or continuity that are maintained correctly.
- Be specific and provide examples from the log chunk.
- Do not list weaknesses or errors, only strengths.

**Output Format:**

**Strengths:**

- [Describe strength 1]

- [Describe strength 2]

**Cons Agent Prompt(Weaknesses Evaluation)=**

**Context:**

You are the Cons Agent in a multi-agent evaluation framework. Your job is to find any problems or weaknesses in a synthetic log that was made. When you evaluate, pay attention to these things:

**Structural errors:** errors such as formatting problems, inconsistent timestamps, or mismatched fields.

**Continuity errors :** problems like logical gaps, missing events, or sequences that do not match with previous chunks.

**Input Provided:**

- Generated log chunk
- Corresponding original anonymized chunk
- Previously generated chunk
- Summaries of the 10 previously generated chunks

**Instructions:**

- List any any mistakes, inconsistencies, or places where the log doesn't follow the format or flow.
- Be clear and include specific examples of the log chunk.
- Do not list strengths of the synthetic log , only weaknesses.

**Output Format:**

**Weaknesses:**

- [Describe weakness 1]

- [Describe weakness 2]

**Judge Agent Prompt (Decision Making) =**

**Context:**

You are the Judge Agent in a multi-agent evaluation framework. Your task is to review the feedback from both the Pros Agent and the Cons Agent and decide whether the generated synthetic log chunk meets the required quality standards.

**Input Provided:**

- Generated log chunk
- Pros Agent strengths list
- Cons Agent weaknesses list
- You may ask the Pros or Cons agents for clarification if needed.

**Instructions:**

- Assess structural fidelity and continuity based on the input of the agents.
- Provide a final pass/fail assessment.
- If the chunk fails, explain why and suggest ways to improve it.
- Keep the evaluation clear and useful.

**Output Format:**

**Structural Assessment:** PASS/FAIL

**Continuity Assessment:** PASS/FAIL

**Overall Assessment:** PASS/FAIL

**Reasoning:** [Explain why]

**Fig. 8:** Prompts for Pros, Cons and Judge agent

## 7.2 Case Study: Multi-Agent Evaluation of a Synthetic Log Chunk

We present a representative example illustrating how the proposed multi-agent evaluation framework assesses a generated synthetic log chunk. The example highlights the practical effectiveness of the proposed evaluation module in validating both structural correctness and contextual continuity of synthetic logs.

**Inputs:**  
*Original anonymized log chunk*  
*Generated synthetic log chunk*  
*Previously generated synthetic chunk*  
*Summaries of the ten previous chunks*

**Original Anonymized Log Chunk:**

*2023-08-14 10:45:12 INFO User\_A logged in from IP\_A on HOST\_X*  
*2023-08-14 10:45:15 ERROR Timeout connecting to SERVICE\_Y*  
*2023-08-14 10:45:20 INFO User\_A retried request to ENDPOINT\_Z*

**Generated Synthetic Log Chunk:**

*2023-08-14 11:02:08 INFO User\_B authenticated from IP\_B on HOST\_M*  
*2023-08-14 11:02:09 INFO User\_B requested resource ENDPOINT\_Q*  
*2023-08-14 11:02:11 ERROR Failed to establish connection with SERVICE\_N*  
*2023-08-14 11:02:12 INFO User\_B reissued request to ENDPOINT\_Q*

**Pros Agent Output**

**Strengths:**  
*The generated chunk preserves the original log structure, including timestamps, log levels, and message formatting.*  
*The error–retry pattern from the original chunk is correctly reproduced in a realistic manner.*  
*The sequence of events is logically ordered and aligns with prior chunk summaries.*

**Cons Agent Output**

**Weaknesses:**  
*The retry occurs immediately after the error, whereas previous chunks typically show a short delay between failure and retry.*

**Judge Agent Decision**  
**Structural Assessment: PASS**  
**Continuity Assessment: PASS**  
**Overall Assessment: PASS**

**Reasoning:**  
*The log chunk maintains correct structure and coherent event flow. Although the retry timing is slightly faster than in previous chunks, this does not break continuity or realism.*

**Recommendations:**  
*No regeneration required.*

**Fig. 9:** Case Study