FAILS: A Framework for Automated Collection and Analysis of LLM Service Incidents

Sándor Battaglini-Fischer*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
s.battaglini-fischer@student.vu.nl

Nishanthi Srinivasan*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
n.srinivasan@student.vu.nl

Bálint László Szarvas*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
b.l.szarvas@student.vu.nl

Xiaoyu Chu†
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
x.chu@vu.nl

Alexandru Iosup†
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.iosup@vu.nl

Abstract

Large Language Model (LLM) services such as ChatGPT, DALL·E, and Cursor have quickly become essential for society, businesses, and individuals, empowering applications such as chatbots, image generation, and code assistance. The complexity of LLM systems makes them prone to failures and affects their reliability and availability, yet their failure patterns are not fully understood, making it an emerging problem. However, there are limited datasets and studies in this area, particularly lacking an open-access tool for analyzing LLM service failures based on incident reports. Addressing these problems, in this work we propose *FAILS*, the first open-sourced framework for incident reports collection and analysis on different LLM services and providers. *FAILS* provides comprehensive data collection, analysis, and visualization capabilities, including: (1) It can automatically collect, clean, and update incident data through its data scraper and processing components; (2) It provides 17 types of failure analysis, allowing users to explore temporal trends of incidents, analyze service reliability metrics, such as Mean Time to Recovery (MTTR) and Mean Time Between Failures (MTBF); (3) It leverages advanced LLM tools to assist in data analysis and interpretation, enabling users to gain observations and insights efficiently. All functions are integrated in the backend, allowing users to easily access them through a web-based frontend interface. *FAILS* supports researchers, engineers, and general users to understand failure patterns and further mitigate operational incidents and outages in LLM services. The framework is publicly available on https://github.com/atlarge-research/FAILS.

CCS Concepts

• **Computer systems organization → Reliability**.

*These authors contributed equally to this research.
†Corresponding authors.

ICPE Companion '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1130-5/2025/05
https://doi.org/10.1145/3680256.3721320

Keywords

failure characterization, LLM, reliability, operational data analytics, incident report, failure recovery, system design

ACM Reference Format:
Sándor Battaglini-Fischer*, Nishanthi Srinivasan*, Bálint László Szarvas*, Xiaoyu Chu†, and Alexandru Iosup†. 2025. FAILS: A Framework for Automated Collection and Analysis of LLM Service Incidents. In *Companion of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE Companion '25), May 5–9, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3680256.3721320

1 Introduction

Large Language Models (LLMs) are transforming industries and societies through a broad range of applications, such as customer service [24], content generation [34], decision-making processes [31], and scientific research [11]. LLM services operate as highly complex distributed systems, involving geographically dispersed components for training, inference, and user interaction [14], which makes them prone to failures in infrastructure, software, and other external dependencies [33]. Failures and outages have major costs for users relying on LLM services, thus can cause significant degradation in customer loyalty, leading even the largest providers to issue a public apology after a system-wide outage [25, 27]. Therefore, understanding and mitigating these failures is a critical challenge to maintain the reliability and trustworthiness of these systems [33].

Currently, tools for analyzing failures in LLM services are either private or have limited functionality. Companies such as SmartBear AlertSite [1], UptimeRobot [3], and Pingdom [2] offer enterprise monitoring services, but are closed-source and aimed at business customers and system administrators. Downdetector [7] offers real-time problem and outage monitoring of public websites, but as a user-driven site lacks comparative tools and in-depth root-cause analysis, as well as the ability to select and compare different services from one provider. Although some work has been done to collect and investigate failures on these pages [16, 28], there is still a lack of tools that integrate data scraping, analysis, and visualization in an open-source framework for researchers, business owners, and the general public.

To address this challenge, we proposed ***FAILS*** *(**F**ramework for **A**nalysis of **I**ncidents on **LL**M **S**ervices)*, which we designed and implemented to: (1) Automatically scrape, clean, and store incident data from LLMs self-disclosed reports; (2) Automatically perform

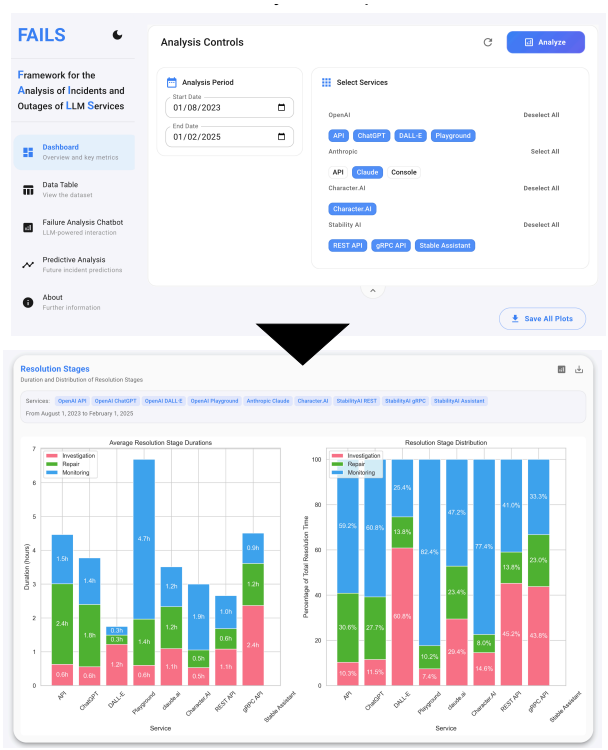


Figure 1: The frontend dashboard of *FAILS* and one of the 17 resulting plots generated from it.

a wide range of failure characterizations, such as distribution of MTTR and MTBF; (3) Visualize data and analysis results, such as temporal failure trends, failure correlations, and recovery patterns; (4) Integrate LLM tools to provide interactions with datasets, and contextual insights into analysis results; (5) Provide a user-friendly and web-based interface to access all functions of the framework (See Figure 1).

Our key contributions are as follows:

- **C1. (Framework Design)** We design *FAILS*, the first tool to automatically collect, monitor, and analyze incident reports from popular LLM providers and services. We describe the architecture and components of the framework, followed by the functional and non-functional requirements we formulated.
- **C2. (Implementation)** We implement *FAILS*, which enables users to select and compare operational characteristics of different services and providers. We provide 17 types of failure analysis that can generate visual plots automatically. We also integrate LLM tools into the framework to allow users to interact with datasets and analyze plots directly in natural languages.
- **C3. (Open Science)** We follow the principles of open science and release the framework on https://github.com/atlarge-research/FAILS.

2 Background

In this section we provide an overview of LLM services, their failure recovery model, and the key concepts and metrics used in this work.

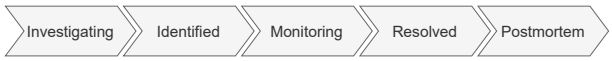


Figure 2: Overview of the failure recovery model [16].

2.1 LLM Services and Incidents

We center our work on 11 LLM services from 4 providers, namely: (1) OpenAI (ChatGPT, API, DALL-E, and Playground), (2) Anthropic (Claude, API, and Console), (3) Character.AI, and (4) Stability.AI (RestAPI, gRPC API, and Stable Assistant). The services are selected because they have publicly accessible incident reports. These incident reports are valuable to understand the LLM availability and their failure recovery patterns.

2.2 Failure Recovery Model and Metrics

The failure recovery analysis of incidents is grounded in the failure recovery model summarized in [16]. This model provides a systematic framework to analyze how providers handle incidents, providing insight into fault tolerance and resilience. The adopted failure recovery model comprises five key statuses that describe the incident handling lifecycle, which is visualized in Figure 2:

- **S1 Investigating:** The operational team begins to analyze the issue immediately after it is observed.
- **S2 Identified:** The problem of the incident is identified.
- **S3 Monitoring:** A fix is implemented and the system is monitoring to ensure the resolution is effective.
- **S4 Resolved:** The incident is considered resolved and normal operations resume.
- **S5 Postmortem:** A detailed summary of the incident is presented to analyze the root cause and suggest measures to prevent recurrence.

This model enables the following key metrics:

MTBF (Mean Time Between Failures): This metric calculates the average time between two consecutive failures for a provider. It serves as an indicator of how fault-tolerant the provider system is.

MTTR (Mean Time To Recovery): Derived from the failure recovery model, MTTR measures the total time taken to resolve a failure, encompassing the period from investigation (S1) to resolution (S4). Provides information on how quickly a provider can respond to and rectify failures.

Co-occurrence of Failures: This metric measures the number of services impacted simultaneously (service inter-dependencies) during a failure for each provider.

3 Design of *FAILS*: a Tool for Automated Incident Data Collection and Analysis of LLM Services

In this section, we first analyze the requirements for *FAILS*. Then, we present its architectural design, including its backend, frontend, and their interactions. Finally, we provide a detailed explanation of the implementation of different components and how they meet the proposed requirements.

3.1 Requirement Analysis

We have identified 5 functional requirements (**FR**), and 4 non-functional requirements (**NFR**) for *FAILS*.

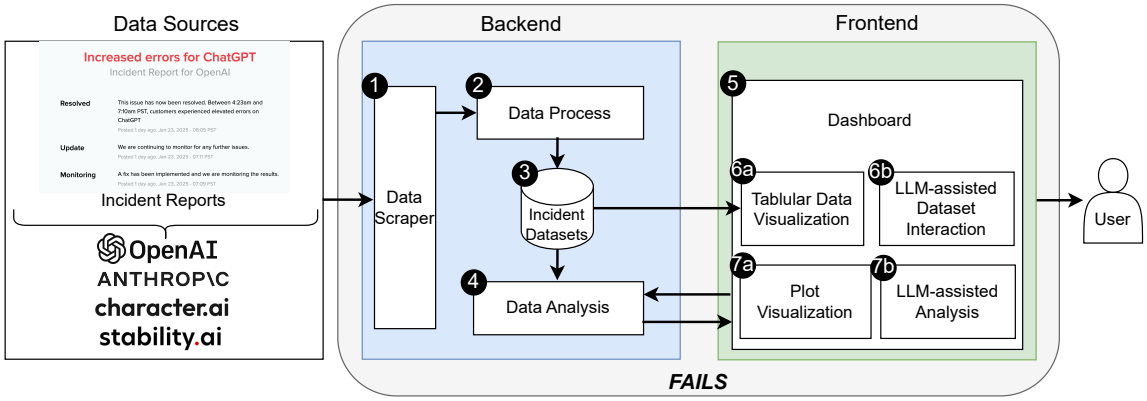


Figure 3: An overview of the architecture of *FAILS*.

FR1 Data Collection: The framework must be able to automatically scrape incident and outage data from multiple service providers. Given the distributed nature of these services, the pipeline must handle potential challenges such as incomplete or changing data structures. The framework should be able to automatically collect, process, and update the datasets according to the user configuration.

FR2 Data Cleaning and Storage: The collected data must be cleaned to ensure that the datasets are consistent and stored in a structured format. Data cleaning includes handling missing values, standardizing date-time formats, and ensuring compatibility between datasets from different providers.

FR3 Failure Recovery Analysis: The framework must be able to analyze the failure recovery process using common-used metrics such as MTTR and MTBF. Failure recovery analysis should allow users to identify failure recovery characteristics, such as evaluating how quickly services recover from failures, and compare failure recovery differences between multiple providers [12, 19].

FR4 Temporal and Co-occurrence Analysis: The framework should support the exploration and comparison of failure trends over time, such as weekly failure distributions, and auto-relations. Additionally, the framework should support the identification of co-occurrence patterns between failures across multiple LLM services.

FR5 Data Visualization and Reporting: The framework should present analytical results in a structured and interpretable manner. This includes generating graphical representations (e.g., time-series plots, heatmaps), tabular summaries, and textual reports that highlight key failure metrics. The focus of this requirement is on ensuring that the results of failure analysis are clearly structured and visually comprehensible for effective interpretation.

NFR1 User-friendly and Accessible Web-based Interface: The framework should provide an intuitive and interactive web-based interface that allows users to configure analyses, explore results, and compare failure patterns effortlessly. Unlike **FR5**, which pertains to how data is structured and represented, this requirement ensures that users, regardless of their technical expertise, can efficiently navigate and interact with the system. A modern, responsive design should support accessibility across different devices and platforms.

NFR2 Reliability and Fault Tolerance: The framework should operate consistently without unexpected delays or errors. In particular, failures during data scraping and analysis could arise from new or inconsistent data formats from different LLM service providers. However, it is unfeasible to account for all errors, so implementing fault tolerance mechanisms, such as retry logic, data validation checks, and user-level error messages, is significant for the robustness and reliability of the system [22].

NFR3 Generality and Scalability: To ensure the generality of our framework, we chose the following LLM providers and services: OpenAI (ChatGPT, API, DALL-E, and Playground), Anthropic (Claude, API, and Console), Character.AI, and Stability.AI (RestAPI, gRPC API, and Stable Assistant). FAILS can be easily scaled to other LLM providers and services if they use similar structured status pages. Additionally, it should be designed to handle growing data volumes and concurrent user access without performance degradation.

NFR4 LLMs-empowered Interaction: Integrating the capabilities of LLM into the framework can provide more contextual insights from the failure patterns observed by the analysis results. LLM empowers users in root cause analysis and helps them understand more technical concepts. For example, the LLM can provide knowledge of the failure models used in the failure recovery analysis.

3.2 Design of *FAILS* Architecture

FAILS follows a modern client-server architecture, consisting of front- and back-end. Figure 3 shows the architecture of the designed framework. In the backend, *FAILS* uses data scrapers (❶) to collect data on outages and incidents separately from the public status pages of OpenAI [8], Anthropic [5], Character.AI [6], Stability.AI [10]. After a series of data cleaning and processing (❷), the raw data are stored as incident datasets (❸). The data analysis (❹) module provides various analysis functions that are called by user requests through the frontend dashboard (❺).

In the frontend, users can select the types of analysis to perform on selected service and provider through a dashboard (5). Tabular data visualization (6a) provides users with an overview of historical incidents, including information on the provider, service, impact, start and end time, and status. To support a better understanding of incident datasets, *FAILS* integrates LLM-assisted dataset interaction (6b), which allows users to inquire about the statistics of the dataset through a chatbot. The results of the analysis are presented through various figures in the visualization of the plot (7a), and enhanced by an LLM-assisted analysis module (7b). This module combines the plot images with predefined instructions to create a prompt to call the LLM API, providing users with a summary and insights from the analysis results.

3.3 Implementation of *FAILS* Components

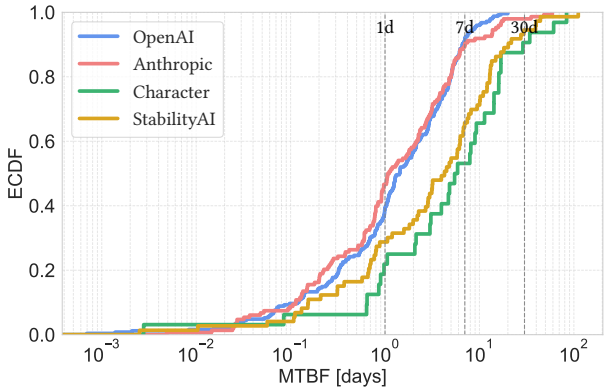
The implementation of the following features addresses the requirements in Section 3.1:

3.3.1 Data scraping and processing pipeline. The data scraping pipeline uses headless Chrome browsers through *Selenium WebDriver* [4] to systematically extract historical outage and incident data from the status pages provided by service providers. It is widely supported, handles complex authorization well, and enables dynamic data retrieval by simulating user interactions, such as clicks and scrolls. The scraper utilizes *WebDriverWait* conditions to ensure proper dynamic content loading and implements retry mechanisms to handle potential stale elements or network problems.

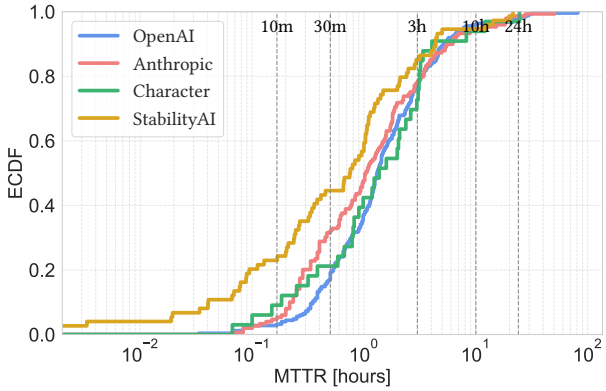
First, attributes and metadata such as title, impact level, and a color that represents severity are extracted from the historical incident list, along with an *ID* to ensure uniqueness. The start and end time are also scraped for the entire failure and, where available, for the individual failure recovery stages (see Section 2.2).

Once extracted, the data undergoes a transformation process, where timestamps are normalized to *UTC*. Service identification is handled through a parsing mechanism that recognizes both explicit service mentions, and implicit references in the incident descriptions. The system then processes and unifies the incident updates, which are stored as *JSON* strings, to categorize the different stages of the incident. The processed dataset is stored in a separate *CSV* file for each pipeline, and then merged after integrity and consistency checks. To prevent data corruption, file backups, temporary files and restore operations are also carried out automatically at this stage. The approach ensures that the pipeline achieves both reliability and scalability, addressing **FR1**, **FR2**, **NFR2**, and **NFR3**.

3.3.2 Tabular data visualization. The CSV format in which the data are saved enables the effective display using a table, such as in *Material-UI's DataGrid* [9]. We used this to display select properties so that users can easily search for information on the latest recorded failures. Through color classifications, users can quickly identify services and impact levels, and read the precise error messages as reported by the providers, all on one page. Sorting by dates, durations, providers, and impact levels helps to provide an overview. It also helps investigate and verify the results of the analysis. In addition, the data scraping pipeline can be started from a button, which automatically updates the table at a fixed time interval. This addresses **FR5**.



(a) Mean Time Between Failures (Longer is better).



(b) Mean Time To Resolve (Shorter is better).

Figure 4: CDF plot of MTBF and MTTR by provider.

3.3.3 Automated plotting. Our "Dashboard" page has the functionality to select a time frame, and a set of individual services from the dataset and plot metrics from this subset of data in an automated layout (addresses **FR3**, **FR4**, and **FR5**). The generated plots can be easily downloaded individually or in bulk as high-resolution PNGs, ready for sharing or use in documents and presentations. The complete list of the plots auto-producted by *FAILS* can be found in Appendix B, Table 2.

3.3.4 LLM-assisted plot analysis. To aid in the interpretation of the generated plots, especially for a nonscientific audience, we implemented an automatic analysis pipeline, which addresses **NFR4**. It enables users to generate an automatic analysis of the current individual plots. This works by sending the image to an LLM service (in our test case, the API of OpenAI's *gpt-4o-mini*), packed in a prompt informed by the knowledge of the individual graph. For example:

> *Analyze this impact level distribution, with a focus on differences in distributions between services. The distribution information:*
> ***{DATE_RANGE, SELECTED_SERVICES, IMAGE}.***
> *The impact levels are defined as* ***{IMPACT_DEFINITION},***
> *The calculated statistical metrics are:*
> ***{STATISTICAL_MEASURES (MEAN, MEDIAN, NUMBER OF INCIDENTS)}.***

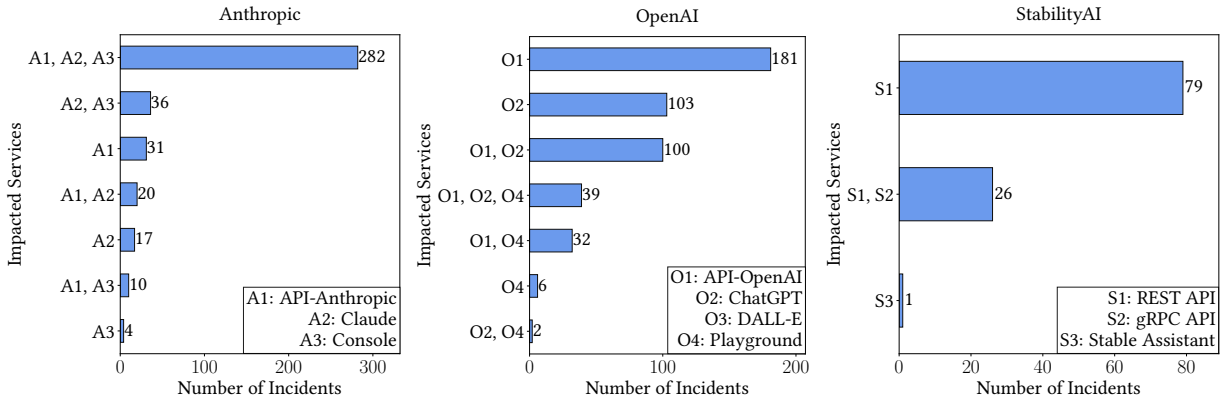


Figure 5: Co-occurrence of failures across services for each provider.

Additional prompts ensure a structured output of a predefined length, which enables us to format and show the response in the front-end. Additionally, we implemented an option to analyze all the plots in bulk, by passing the entire set of generated plots with a prompt, telling the model to focus on the most significant findings.

3.3.5 Interacting with the dataset using LLMs. To add a more dynamic way of interacting with the data (**NFR4**), we input a cleaned version of the dataset and its metrics into the LLM API. In this way, users can interact with a chatbot interface to ask targeted questions about the dataset, such as the number of incidents, the time range of the dataset, and the root causes of individual events. Especially in conjunction with the data table and the generated plots, this enables deep insight into the data. Further experimentation with different prompting paradigms could result in a more informative output from the LLM [30].

3.3.6 Functionality wrapped in a web interface. All of this functionality is neatly wrapped in a web-app interface (**NFR1**), which includes modern functionality such as theming and responsiveness. The individual functions mentioned above are arranged in a sidebar menu, which is clean and easily accessible and includes a page with information on the project.

4 Experimental Results

In this section, we select 3 showcases to demonstrate *FAILS*'s functions, including: (1) The collected incident dataset by *FAILS*; (2) We give 2 types of analysis as examples: MTBF and MTTR distributions, and co-occurrence of failures; (3) The dataset interaction through a LLM-enabled chatbot.

4.1 LLM Incidents Dataset Collected by *FAILS*

To prepare the dataset used for failure analysis, we scraped and processed the data from the selected providers' status pages using *FAILS*, according to the process described in section Section 3.3.1. The resulting dataset includes all history incident reports provided by the LLM providers, starting from the date the first incident was reported until the day of scraping (2025-01-10 13:03:25).

The collected LLM incidents dataset is summarized in Table 1. The incident reports span a time range from approximately 11 months (Stability.AI) to nearly 4 years (OpenAI), based on the release dates of their services.

Table 1: Summary of the LLM incident dataset collected on 2025-01-10.

Provider	First Date	Last Date	Number of Reports
OpenAI	2021-02-09	2025-01-08	414
Anthropic	2023-03-16	2025-01-09	225
Stability.AI	2023-01-24	2024-12-19	94
Character.AI	2023-10-24	2025-01-10	59

4.2 Failure Analysis of LLMs Service Incidents

We present the following failure analysis results, which are generated automatically by *FAILS*.

4.2.1 MTBF and MTTR. Figure 4a displays the Cumulative Distribution Function (CDF) of the MTBF in days for each provider. A steeper CDF indicates that MTBF values are less evenly distributed, while a distribution function shifted further to the right signifies longer times between failures for the provider.

Notably, Character.AI and Stability.AI exhibit longer MTBFs and less steep CDFs compared to OpenAI and Anthropic, which is likely due to the higher complexity and user demand faced by the latter two providers. In contrast, smaller user bases and simpler system designs likely contribute to the extended MTBFs of Character.AI and Stability.AI.

The CDF distributions of the MTTR in hours for each provider are presented in Figure 4b. A steeper CDF indicates less evenly distributed MTTR values, while a distribution function shifted further to the right represents longer recovery times for the provider. Stability.AI exhibits a left-shifted CDF, indicating significantly shorter recovery times than OpenAI, Anthropic, and Character.AI.

4.2.2 Co-occurrence of Failures. Figure 5 highlights the number of services affected simultaneously during failures for each provider. The subplots illustrate the number of services impacted during a failure: (a) For Anthropic, failures most frequently impact all three services simultaneously; (b) For OpenAI, failures tend to impact a single service more often than multiple services; (c) For Stability.AI, failures predominantly affect only one service at a time.

In summary, Anthropic frequently experiences failures affecting multiple services, suggesting significant inter-dependencies among its systems. In contrast, OpenAI and Stability.AI tend to

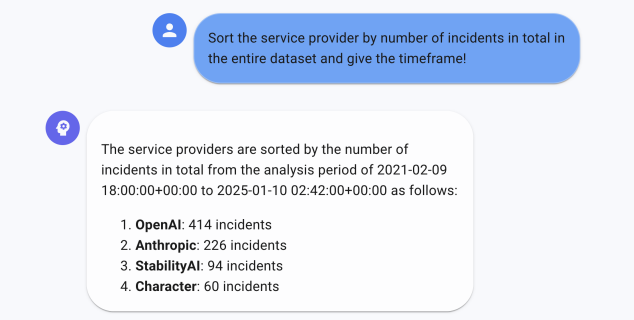


Figure 6: The chatbot in *FAILS*. Here users can interact with the dataset in natural language.

experience failures that impact only one service, indicating better compartmentalization and reduced cascading risks.

4.3 Dataset Interaction through LLM Chatbot

The LLM-assisted analysis feature (6b in Figure 3), accessible from the sidebar of the frontend dashboard, enables users to query the dataset through a chatbot using natural language in a conversational format. Figure 6 shows an example use case, where the user asked the chatbot about the time range and total number of incidents. The chatbot responds based on the provided dataset, answering the user's questions.

However, the accuracy and quality of the response depend on the chosen LLM, the prompts used, and are constrained by the LLM's limitations. This limitation of our work will be further discussed in Section 5.

5 Limitations and Validity

Our framework provides a foundation for effective analysis of LLM service failures. However, our approach has certain limitations:

The incompleteness of datasets: Despite gathering incident data from four LLM providers and eleven services, the fact that these data are self-reported by operators poses risks of inaccuracy or incomplete disclosure. Some incidents may occur without being detected by the operational team. Moreover, companies may choose not to disclose all incidents due to potential impacts on market interests and policies.

The accuracy of LLM-assisted tools: For the interaction and analysis of LLM-assisted datasets, the accuracy of LLM-generated content is not explicitly evaluated. As a result, the reliability of the output solely depends on the performance of the LLM APIs, raising concerns that it may not accurately reflect the characteristics of the data and plots.

6 Related Work

In the relatively new field of LLMs, previous work has begun to address the reliability of these public services by analyzing their outages and failure-recovery processes. For example, previous wrok [16] conducted an empirical characterization of outages and incidents in public LLM services, providing valuable insights into metrics such as MTBF and MTTR as well as temporal patterns and co-occurrences. While this work sets the foundation for our work, it focuses primarily on establishing the metrics needed for the analysis without offering a practical, interactive, and extendable tool for further exploration or visualization of such data.

In the broader context of failure characterization, previous work has extensively explored failures across various related domains, such as HPC [18, 26], machine learning [15, 17], cloud computing [13], and other large-scale distributed systems [20, 32]. While these studies have improved fault tolerance mechanisms, they are not directly tailored to the operational challenges and complexities of LLM services. In-domain studies such as the dynamic of the response and requests [29], training [21], and inference [23] focus primarily on the performance and scalability aspects of LLM workloads rather than the characterization of failures and recovery processes.

Our framework fills the gap in the current corpus of work on failure analysis of LLM services by providing an open-source tool for researches that combines well-known failure characterization metrics with a modern, LLM assisted approach.

7 Conclusion and Future Work

FAILS enhances failure analysis in complex LLM ecosystems by integrating automated data scraping, advanced analytics, and intuitive visualizations. With real-time data integration, predictive modeling, and geographic user-reported incident analysis, it helps improve reliability and decision-making in large-scale AI systems.

The *FAILS* streamlines failure analysis by integrating data processing, visualization, and analytics. Its robust scraping pipeline ensures reliable, up-to-date data across providers, enabling detailed studies of MTBF, MTTR, and failure co-occurrence patterns. The interactive dashboard allows users to explore failure trends dynamically. Automated CDF visualizations highlight recovery time variations, while LLM-assisted analysis provides deeper insights into provider-specific resilience. The intuitive design of the framework enables rapid visualization adjustments, allowing both technical and non-technical users to easily conduct meaningful comparisons. By reducing manual effort and enhancing cross-provider comparisons, *FAILS* provides actionable insights that contribute to improving system resilience and reliability.

Our future work includes: (1) Real-time prediction: Enhancing *FAILS* with real-time failure tracking will open new possibilities, such as predicting recovery times upon the first report of downtime. By modeling provider, service, location, and historical data as a feature vector, a transformer-based approach can improve response strategies. (2) RAG-enhanced analysis: Strengthening LLM-driven insights through refined prompts and a *Retrieval Augmented Generation (RAG)* system will enhance contextual understanding. Integrating local models instead of APIs will improve fault tolerance, ensuring *FAILS* remains fully functional even during service disruptions. (3) Comparison with user reports: Extending the failure analysis with user-reported data from third-party platforms (e.g. Downdetector or X/Twitter) will provide a more comprehensive view of incidents. Advancing our approach to dynamic graph scraping will enable deeper comparisons between provider and user-reported failures, improving the data transparency and accuracy.

References

[1] 2018. AlertSite Software as a Service - Everbridge at SmartBear Connect. https://smartbear.com/product/alertsite [Accessed 14. Jan. 2025].

[2] 2024. Pingdom. https://www.pingdom.com [Accessed 14. Jan. 2025].

[3] 2024. UptimeRobot: Free Website Monitoring Service. https://uptimerobot.com [Accessed 14. Jan. 2025].

[4] 2025. About Selenium. https://www.selenium.dev/about [Accessed 14. Jan. 2025].

[5] 2025. Anthropic Status. https://status.anthropic.com [Accessed 13. Jan. 2025].

[6] 2025. Character AI Status Status. https://status.character.ai [Accessed 13. Jan. 2025].

[7] 2025. Downdetector. https://downdetector.com [Accessed 14. Jan. 2025].

[8] 2025. OpenAI Status. https://status.openai.com [Accessed 13. Jan. 2025].

[9] 2025. React Data Grid component - MUI X. https://mui.com/x/react-data-grid/?srsltid=AfmBOoozaD7Vh9WjEftiJHA_6H7zcC6IxiDtbtqjVAlI_JDkBaDBdTsi [Accessed 13. Jan. 2025].

[10] 2025. Stability AI Platform - Status. https://stabilityai.instatus.com [Accessed 13. Jan. 2025].

[11] Md Arman and Umama Lamiyar. 2023. Exploring the Implication of ChatGPT AI for Business: Efficiency and Challenges. *International Journal of Marketing and Digital Creative* 1 (Sept. 2023), 64–84. doi:10.31098/ijmadic.v1i2.1872

[12] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing* 1, 1 (2004), 11–33.

[13] Mehmet Berk Cetin, Sacheendra Talluri, and Alexandru Iosup. 2021. Characterizing User and Provider Reported Cloud Failures. *arXiv* (Oct. 2021). doi:10.48550/arXiv.2110.12237 arXiv:2110.12237

[14] Zina Chkirbene, Ridha Hamila, Ala Gouissem, and Unal Devrim. 2024. Large Language Models (LLM) in Industry: A Survey of Applications, Challenges, and Trends. In *2024 IEEE 21st International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*. IEEE, 229–234.

[15] Xiaoyu Chu, Daniel Hofstätter, Shashikant Ilager, Sacheendra Talluri, Duncan Kampert, Damian Podareanu, Dmitry Duplyakin, Ivona Brandic, and Alexandru Iosup. 2024. Generic and ML Workloads in an HPC Datacenter: Node Energy, Job Failures, and Node-Job Analysis. In *2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS)*. 710–719. doi:10.1109/ICPADS63350.2024.00097

[16] Xiaoyu Chu, Sacheendra Talluri, Qingxian Lu, and Alexandru Iosup. 2025. An Empirical Characterization of Outages and Incidents in Public Services for Large Language Models. arXiv:2501.12469 [cs.PF] https://arxiv.org/abs/2501.12469

[17] Xiaoyu Chu, Sacheendra Talluri, Laurens Versluis, and Alexandru Iosup. 2023. How Do ML Jobs Fail in Datacenters? Analysis of a Long-Term Dataset from an HPC Cluster. In *Proceedings of the International Conference on Performance Engineering, Coimbra, Portugal, April, 2023*.

[18] Sheng Di, Hanqi Guo, Eric Pershey, Marc Snir, and Franck Cappello. 2019. Characterizing and Understanding HPC Job Failures Over The 2K-Day Life of IBM BlueGene/Q System. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 473–484. doi:10.1109/DSN.2019.00055

[19] Marc Gamell, Daniel S Katz, Hemanth Kolla, Jacqueline Chen, Scott Klasky, and Manish Parashar. 2014. Exploring automatic, online failure recovery for scientific applications at extreme scales. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 895–906.

[20] Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. 2017. Failures in large scale systems: long-term measurement, analysis, and implications. In *ACM Conferences*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3126908.3126937

[21] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, Yonggang Wen, and Tianwei Zhang. 2024. Characterization of Large Language Model Development in the Datacenter. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 709–729. https://www.usenix.org/conference/nsdi24/presentation/hu

[22] Sucharitha Isukapalli and Satish Narayana Srirama. 2024. A systematic survey on fault-tolerant solutions for distributed data analytics: Taxonomy, comparison, and future directions. *Computer Science Review* 53 (2024), 100660.

[23] Małgorzata Łazuka, Andreea Anghel, and Thomas Parnell. 2024. LLM-Pilot: Characterize and Optimize Performance of your LLM Inference Services. *arXiv* (Oct. 2024). doi:10.48550/arXiv.2410.02425 arXiv:2410.02425

[24] Sudeep Meduri. 2024. Revolutionizing Customer Service : The Impact of Large Language Models on Chatbot Performance. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 10 (Oct. 2024), 721–730. doi:10.32628/CSEIT241051057

[25] @OpenAI. 2024. OpenAI's post on X apologizing for downtime. https://x.com/OpenAI/status/1867000372826607627. Accessed: 2025-01-15.

[26] Ju-Won Park, Xin Huang, and Chul-Ho Lee. 2023. Analyzing and predicting job failures from HPC system log. *J. Supercomput.* 80, 1 (June 2023), 435–462. doi:10.1007/s11227-023-05482-y

[27] New Relic. 2023. Observability Report. https://newrelic.com/resources/report/observability-forecast/2023/about-this-report. Accessed: 2025-01-15.

[28] Sacheendra Talluri, Leon Overweel, Laurens Versluis, Animesh Trivedi, and Alexandru Iosup. 2021. Empirical Characterization of User Reports about Cloud Failures. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. 158–163. doi:10.1109/ACSOS52086.2021.00039

[29] Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. 2024. BurstGPT: A Real-world Workload Dataset to Optimize LLM Serving Systems. *arXiv* (Jan. 2024). doi:10.48550/arXiv.2401.17644 arXiv:2401.17644

[30] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* (2023).

[31] Weichao Xu, Huaxin Pei, Jingxuan Yang, Yuchen Shi, Yi Zhang, and Qianchuan Zhao. 2024. Exploring Critical Testing Scenarios for Decision-Making Policies: An LLM Approach. *arXiv* (Dec. 2024). doi:10.48550/arXiv.2412.06684 arXiv:2412.06684

[32] Nezih Yigitbasi, Matthieu Gallet, Derrick Kondo, Alexandru Iosup, and D. Epema. 2010. Analysis and Modeling of Time-Correlated Failures in Large-Scale Distributed Systems. *Telecommunications Policy - TELECOMMUN POLICY* (Oct. 2010), 65–72. doi:10.1109/GRID.2010.5697961

[33] Guangba Yu, Gou Tan, Haojia Huang, Zhenyu Zhang, Pengfei Chen, Roberto Natella, and Zibin Zheng. 2024. A Survey on Failure Analysis and Fault Injection in AI Systems. *arXiv* (June 2024). doi:10.48550/arXiv.2407.00125 arXiv:2407.00125

[34] Pengyuan Zhou, Lin Wang, Zhi Liu, Yanbin Hao, Pan Hui, Sasu Tarkoma, and Jussi Kangasharju. 2024. A Survey on Generative AI and LLM for Video Generation, Understanding, and Streaming. *arXiv* (Jan. 2024). doi:10.48550/arXiv.2404.16038 arXiv:2404.16038

A Screenshots from *FAILS*

Figure 7 shows the tabular data view (6a in Figure 3) and Figure 8 shows the long context interaction with the dataset through LLM chatbot (6b in Figure 3).

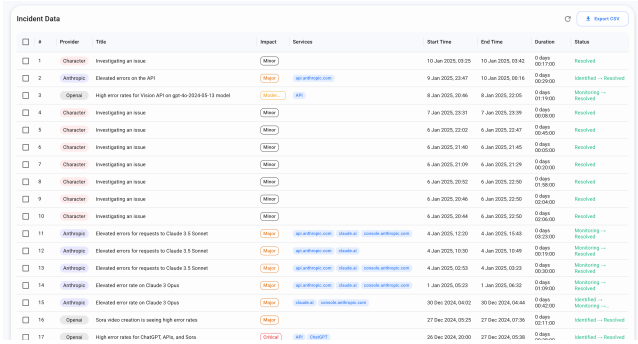


Figure 7: The tabular dataset view in *FAILS*. It allows researchers to view each incident in a sorted format.

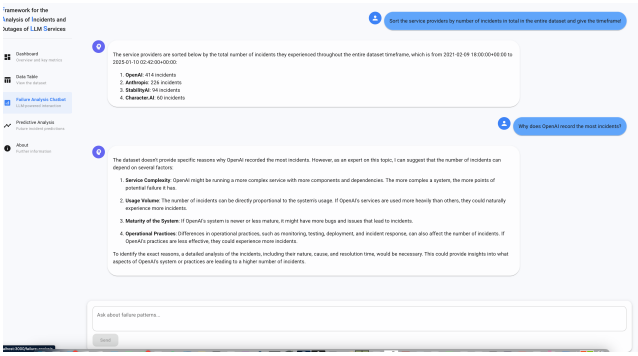


Figure 8: The chatbot in *FAILS*. Here users can interact with the dataset in natural language.

B List of Plots

Table 2 summarizes all the plots that are currently functional in *FAILS* (4 and 7a in Figure 3).

Table 2: The complete list of failure analysis types that *FAILS* can perform and visualize.

#	Plot Name	Description
1	Weekly Overview	Incident distribution across services by day of the week.
2	Hourly Overview	Incident distribution across services by hour of the day (UTC).
3	MTTR Distribution	Recovery time duration distributions for each service.
4	MTTR by Provider	Recovery time duration grouped by provider.
5	MTTR Boxplot	Boxplots showing MTTR statistics per service.
6	MTBF Distribution	Time-between-failures distributions for each service.
7	MTBF by Provider	Time-between-failures grouped by provider.
8	MTBF Boxplot	Boxplots showing MTBF statistics per service.
9	Resolution Activities	Resolution process duration and distribution.
10	Status Combinations	Distribution of status combinations.
11	Daily Availability	Day-by-day service uptime analysis.
12	Service Co-occurrence	Correlation matrix of co-failing services.
13	Co-occurrence Probability	Probability matrix of concurrent failures.
14	Service Incidents	Breakdown of incidents by service.
15	Incident Outage Timeline	Timeline of service disruptions.
16	Autocorrelations	Time-series analysis of incident correlations.
17	Incident Impact Distribution	Breakdown of incident severity by provider.