Exploration on Application Health Monitoring and Marketplace

Kavin Manimaran[1], Ryan Lee Jun Wei[2], Zhang Chenghai Christopher[3]
[1]Anglo Chinese School (Independent), 121 Dover Road, Singapore 139650
[2]Victoria School, 2 Siglap Link, 448880
[3]Defence Science Technology Agency, 1 Depot Road, Singapore 109679

1. Abstract

In today's fast-paced and technology-driven world, site reliability engineers face an overwhelming challenge: sifting through vast amounts of log data to identify and resolve system issues. As the development of IT becomes increasingly complex, manually inspecting lines of logs to pinpoint the various bugs and errors to find the root cause of problems is not only time-consuming but also prone to human error. This type of process is extremely tedious and is overly taxing on the human body and capability. This process can further lead to delays in troubleshooting, impacting business operations and customer satisfaction.

In order to tackle such a problem, we believe that with the assistance of AI, for example, OpenAI's ChatGPT, this problem can be solved through the help of technology. Artificial Intelligence is very well known for efficiently replacing the job of humans to quickly and more accurately complete the task. With the nature of a site reliability engineers' job, AI can offer a massive role in helping to reduce the painstaking effort of going through thousands of lines of logs.

AI can analyze log data to detect patterns, anomalies, and bugs that may go unnoticed by human analysts. AI-driven tools can automate the identification of issues, quickly isolate the root causes, and even provide actionable insights or recommended solutions. This reduces the burden on IT analysts, allowing them to focus on more strategic tasks, improving system uptime, and enhancing overall operational efficiency. In essence, AI can empower IT teams to work smarter, not harder, by providing faster and more accurate insights into system health.

2. Aim

The goal of this project is to showcase the use of AI-driven tools to help analyse error logs in the daily work of an enterprise IT analyst and provide helpful solutions for troubleshooting identified issues.

3. Problem

As of today, most site reliability engineers manually read through lines of logs to identify errors within a system and have to think of solutions to troubleshoot in on the spot. Despite advancements through usage of observability platforms such as Dynatrace, troubleshooting and issue identification are still conducted manually using rule-based filtering logic to detect potential root causes.

4. Introduction

4.1 OpenAI

OpenAI, founded in December 2015, emerged with the mission of advancing digital intelligence in a way that benefits humanity as a whole. Established by a group of tech visionaries—including Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, and John Schulman—OpenAI was originally conceived as a non-profit organization. [1] OpenAI transitioned into a "capped-profit" company in 2019, a structure designed to attract significant investment while upholding its commitment to prioritize safety and ethical considerations. This shift allowed OpenAI to scale its research and operations, leading to significant technological milestones such as the development of GPT-2, GPT-3, and later, GPT-4. GPT's Large Language model has made it extremely suitable for this research. Leveraging NLP, Natural Language Processing, GPT comprehends the context and language used in log messages. This enables the identification of keywords and phrases that signify an issue and provide actionable insights on how to address them efficiently. GPT's LLMs are trained on large amounts of information which allows them to give accurate results based on a probability. For example, GPT-2 was trained on a massive dataset of texts and codes of approximately 40GB in size and 1.5 billion parameters. [2]

4.2 Site Reliability Engineers & Log Data

Site reliability engineers are essential to the design, development, and maintenance of complex systems. They act as the backbone of multidisciplinary teams, integrating various subsystems into a cohesive and functional whole. [3] Site reliability engineers focus on creating comprehensive solutions that address both technical specifications and business requirements. Their primary responsibilities include defining system requirements, designing architectures, and ensuring system performance through testing and validation. Site reliability engineers play a pivotal role in the lifecycle of a system, ensuring it meets the desired objectives efficiently and effectively. [4]

Their work involves a deep understanding of system modeling and analysis, using tools such as SysML (Systems Modeling Language) for visualizing system components and behaviors. In addition, site reliability engineers are skilled in managing projects and collaborating with stakeholders, balancing technical expertise with strategic oversight. This cross-functional role requires strong problem-solving skills and the ability to translate complex technical language into actionable plans for both technical and non-technical audiences. [5]

Log files are an integral part of site reliability engineering, playing a crucial role in monitoring, troubleshooting, and maintaining system health. The files capture detailed records of system events, errors, and operations, providing vital insights for diagnosing issues. Site reliability engineers often rely on these logs to identify the root causes of performance issues, security breaches, or operational failures. The process of sifting through extensive logs can be time-consuming and labor-intensive. According to research, the average site reliability engineer spends significant time-up to 30-40% of their working day-analyzing log data to maintain system reliability. [6]

The challenge of parsing through large volumes of log data to identify critical information is substantial. With systems generating terabytes of data daily, manual log analysis becomes impractical. This can lead to delays in problem detection and resolution, potentially affecting system performance and uptime. The need for efficient tools that can help site reliability engineers streamline the log analysis process is therefore crucial. Automated solutions can reduce the time spent on these tasks, significantly enhancing productivity and system reliability. [7]

The real value of generative AI, particularly GPT models, lies in its ability to automate the analysis of large datasets or logs. Tasks that would traditionally take hours, such as identifying patterns or anomalies in log files, can now be accomplished in minutes using GPT models in an automated workflow. This reduces the manual labor involved in troubleshooting and improves efficiency. By automating these processes, engineers can focus on more complex tasks, ensuring quicker problem resolution and higher productivity.

In our project, while we will leverage the OpenAI Playground for initial testing and to fine-tune parameters such as temperature, top-p, and frequency, our focus will be on using these settings to develop an automated solution. The goal is to optimize the model for site reliability engineers to resolve issues more quickly and with greater efficiency. These adjustments will help tailor the model's behavior to meet the specific needs of automated troubleshooting, ensuring optimal results. Further details on how we fine-tune these parameters will be discussed later in this report. [9]

4.3 Generative AI

Generative AI is a branch of artificial intelligence focused on creating new content, such as text, images, audio, and video, by learning from existing data. It utilizes advanced algorithms and deep learning models to generate human-like or original material that closely resembles the examples it was trained on. Generative AI is powered by models like transformers, generative adversarial networks (GANs), and diffusion models. These technologies have revolutionized content creation, enabling the production of high-quality, realistic content in various forms. [10]

In particular, Large Language Models (LLMs), a subset of generative AI, are designed to understand and generate human-like text. They use transformer architectures to process and generate language-based content with remarkable fluency and coherence. These models, such as GPT (Generative Pre-trained Transformer), learn from vast datasets containing a wide array of texts, allowing them to perform a variety of language tasks, including writing articles, answering questions, and even mimicking specific writing styles. LLMs work by capturing the relationships between words and phrases, enabling them to generate contextually relevant text. This makes them incredibly powerful for applications in customer service, content creation, research, and more. [11]

Generative AI, particularly through LLMs, has greatly impacted industries such as content creation, automated coding, music composition, and even film production. By automating and enhancing creative processes, generative AI provides powerful tools that enable more efficient workflows and open new possibilities for innovation across various fields. [12]

5. Materials & Methods

5.1 Materials

While doing this research, we mainly used the easyTravel demo application from Dyntrace as a reference to what a real enterprise's system logs may look like and relied on ChatGPT as our LLM model to analyze errors on the engineer's behalf.

easyTravel simulates an e-commerce travel portal where users can log in, search for journeys to various destinations, select available promotional journeys, and book their trips using credit card details. An app launcher is provided to allow user to conveniently switch between different demo scenarios to simulate different error types. Each scenario can be modified or extended by updating an XML scenario file. This is useful when giving demos and allows you to focus on problem areas that are particularly relevant for a specific demo. [13]

Using easyTravel, we were able to explore the working of an enterprise system and create many different scenarios to replicate the issues a site reliability engineer might be facing on a day-to-day basis on the job. By experiencing their job firsthand, we are able to better empathise and brainstorm ideas to ease the burden of their job. We examined the various types of log files that a site reliability engineer typically analyses to identify the root causes of errors and troubleshoot effectively.

As we put ourselves in the shoes of a site reliability engineer, we recognized the difficulties of the job. Manually sifting through the thousands of lines in log files to troubleshoot and debug is a painful and tiring process. We realised that this task can possibly be augmented using AI while still ensuring accuracy and efficiency. Hence, this led us to our proposed solution that we will be covering later in this report.

With the endless possibilities with programming and various technologies, we thought of writing a program specifically targeted to help a site reliability engineer sift through their endless log lines. With the help of experts online and AI, we think it is possible to create such a program capable of instantly returning only important error messages from an entire file of thousands of lines in the span of seconds. This significantly reduces the time and effort required from a site reliability engineer, enabling them to focus their energy on other important tasks.

```python
def filter_error_logs(log_file_path):
    try:
        with open(log_file_path, 'r', encoding='utf-8', errors='ignore') as log_file:
            for line in log_file:
                if 'ERROR' in line:
                    print(line.strip())  # Print the error line without extra spaces
    except FileNotFoundError:
        print(f"Error: The file '{log_file_path}' was not found.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

log_file_path = r"C:\Users\UserAdmin\.dynaTrace\easyTravel 2.0.0\easyTravel\log\BusinessBackend.log"
filter_error_logs(log_file_path)
```

The picture above showcases the codes we used to filter the log files. By copying the path of the log file that we are interested in filtering, we can paste the path into the log_file_path variable to carry out the function created which will print out only the error messages from the entire log file. As such, we can focus on the more important information instead of looking at the other logs.

Firstly, we defined a function for filtering the log files so that we can easily call this function. Next, we set the parameters as the log file path so that we can open the given file. The next few lines call Python to run through each line of logs in the log file and check for the word “ERROR” which indicates that the log data requires immediate attention. If the line meets our condition, we classify it as an error log and then print it in our output.

From 1 log file → 8670 lines of logs
After filtering → 232 lines of error logs

This data shows how useful the filtering program can be as it reduces the workload of an engineer to go through the many lines of logs but instead only needing to focus on each line of the new filtered data.

Through our ideation and brainstorming, we directed our attention to the use of ChatGPT’s playground to help with the task of analysing and providing solutions to resolve error logs. ChatGPT offers advanced features such as allowing users to specify output formats and customize tool parameters. With such features, we can configure ChatGPT to play the role of a Site Reliability Engineer’s advisor. This approach allows us to customise the responses to be more relevant, as the AI model generates them from the perspective of a site reliability engineer, ensuring the output response is ideally suited for the role. As a site reliability engineer, the main goal is to find the root of the issue and troubleshoot it as quickly as possible. As such, we can instruct the API to keep its answers short and straight to the point for maximum efficiency while maintaining high accuracy. It is then instructed to give step by step instruction on solving the issue at hand which can be helpful to an engineer. We provided a detailed and specific prompt that can maximise the use of the API to return only necessary and important information.

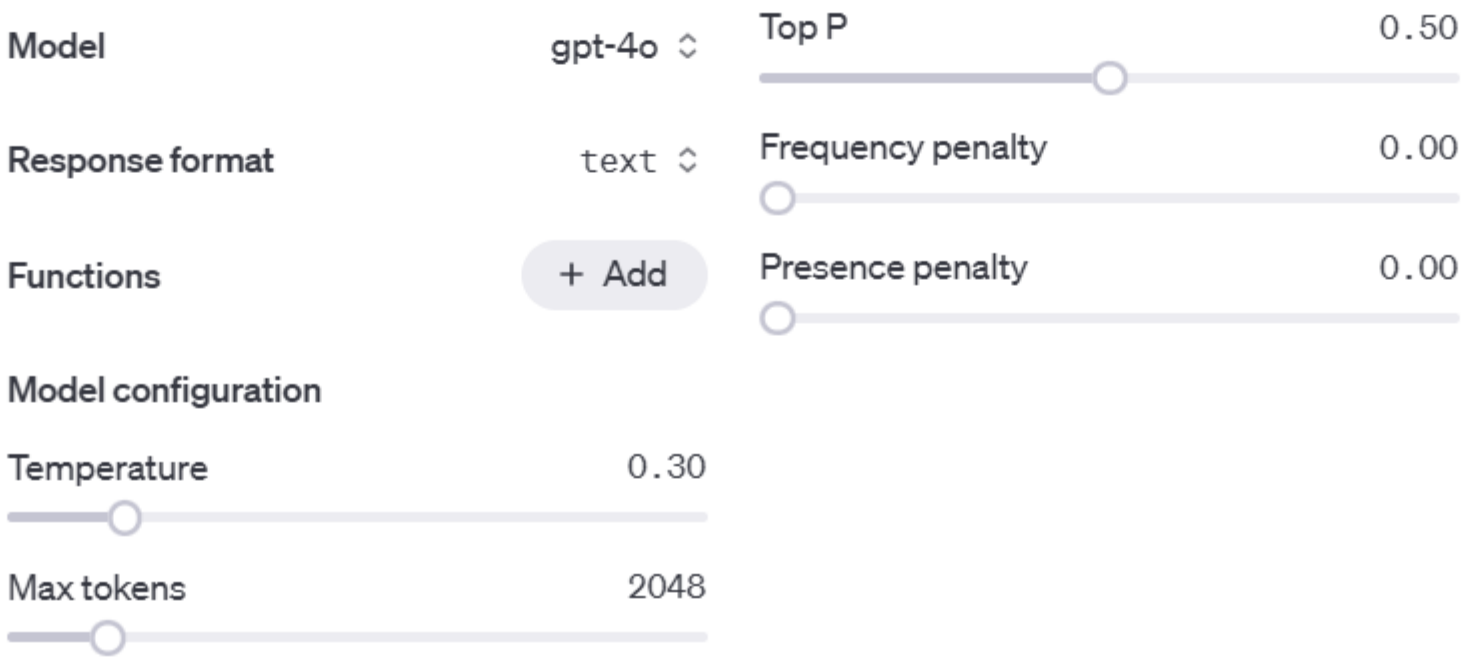
Furthermore, ChatGPT Playground allows room for further customisation for a specific response type. There are mainly 5 options that we will be looking into: Temperature, Max Token, Top P, Frequency Penalty and Presence Penalty. [14]

1. Temperature
    a. **Temperature** is a parameter that controls the level of randomness or creativity in the model's responses. It affects how deterministic or varied the model's output is. It ranges from 0 to 2 with 0 being not random and 2 being extremely varied and broad
2. Max Token
    a. In the ChatGPT Playground, the **max tokens** parameter controls the maximum number of tokens (words or pieces of words) that the model can process and generate in a single interaction. This includes both the input and the output.
    b. $0.03 for every 1,000 input tokens and $0.06 per 1,000 output tokens
3. Top P
    a. The **top_p** parameter is used to control the diversity and creativity of the model's responses. It's part of the nucleus sampling method for generating text, which is an alternative to temperature. **Top-p** selects a subset of tokens that together make up a certain probability mass and then samples from that subset, directly limiting the pool of choices.
4. Frequency Penalty
    a. The **frequency penalty** is a parameter used to penalize the model for generating repetitive content. It helps reduce the likelihood that the model will repeat the same phrases or words within a single response, encouraging more diverse and varied outputs.
5. Presence Penalty

a. **The presence penalty** parameter adjusts how likely the model is to talk about topics it has already mentioned. It is a parameter designed to encourage more diverse and dynamic responses by discouraging the model from repeating itself or sticking too closely to previous themes in the conversation

6. Results and Discussion

6.1 OpenAI playground's configuration



Below are the permutations done by us through many rounds of testing and observations.

1) **Temperature**
   a) Considering the nature of our project, we would prefer a direct and less varied response as we are trying to be as accurate as possible to minimize errors and to ensure a longer server uptime. As such, through testing the result of changing the temperature we found various results. Firstly, from the 0.7-1.2 range, the answers are very standard and gives a well-balanced answer that is normally expected from an AI. However, past the 1.2 range is where the answers start to get adventurous and very broad. For instance, when maxed out at 2.0, the answers produced even gave varying languages. Though being imaginative and creative is useful, our aim of this research does not require such imagination but instead the opposite. Furthermore, with such varying answers, the output tokens will increase significantly to incorporate all of the different ideas. Therefore, we believe that a good range for this project would be from the 0-0.3 range. This range gives a short and direct answer which is exactly what we require from the API to be of good assistance to an engineer, anything higher being too long and instead slowing them down
2) **Max Token**
   a) In a realistic and practical scenario, operational expenses are necessary to maintain the use of an API that assists a site reliability engineer in their daily tasks. Furthermore, we will not be using the API only once or twice but maybe hundreds or maybe thousands of times a day as errors can come up at any time. As such, we have to set a limit to the max tokens as we do not want to waste unnecessary money when we can save cost while still ensuring a suitable performance. Since we already want the answer to be short and direct, by reducing the max token we are

making sure that ChatGPT does not provide us with extra or unnecessary information that will not be helpful for an engineer's task.

b) However, we realised when configuring the max tokens that although we can save cost by reducing the max tokens, the output ends abruptly without being able to complete its sentence majority of the time. This is due to the nature of the configuration. Once the output generated hits the max token, the entire processing will stop and will not consider the overall outcome. Furthermore, while testing out various questions, we realised that without a word limit, ChatGPT will elaborate on extra information that is not needed as it is only a waste of our tokens. Therefore, in order to tackle this problem we decided to keep a word limit in the system message. By doing so, we are able to ensure that the information will not end abruptly while still ensuring an accurate response as now OpenAI will consider the word limit before generating the answer. Through varying tests, we decided that a range of 50-100 words will be sufficient for any given prompt of the nature of our project by giving a short and simple summary as well as the troubleshooting methods. Anything below will end up with a short answer not giving enough detail and information while anything above being too long winded and unnecessary.

3) **Top-P**

a) While Top-p is of a similar nature to temperature by controlling the randomness, they work hand in hand to give an answer type a user wants. Since we are targeting simple and direct answers, we believe that keeping the Top-p value on the lower range will benefit us better as it will make the answer more coherent and focused. Our basis for choosing the range is also similar to Temperature where 1.0 being a bit too broad for our project with answers being accurate yet unnecessary. Somewhere in the middle range of 0.5 is what meets our goal of being direct and straightforward.

4) **Frequency Penalty**

a) As a site reliability engineer's work is very technical and has very little room for creativity, it is better for the API to use similar words or phrases as it makes the results easier to understand to a site reliability engineer as well as keeping the entire system systematic. This led us to set 0 penalty for repeating similar words or phrases as we want the API to maintain a standard consistent answering style

5) **Presence Penalty**

a) Since we are trying to stick to a consistent theme of analysing and troubleshooting logs, we believe that it is unnecessary to put a penalty on the AI as we want to encourage to repeat itself as we believe that some troubleshooting methods are very specific and there is not much creativity options

6.2) System Message for OpenAI

System message Generate

1. You are a system engineer
2. You are to analyse error log details and understand it
3. Summarise the information and provide an indepth step-by-step troubleshooting method for the error
4. Keep it short and simple, easy to understand
5. Be sure to consider different potential factors that could lead to the error and how to address them.
6. Keep everything under 100 words.

Above shows how we customized OpenAI to take the role of a site reliability engineer to better answer the prompt it is given in context to our project. By doing so, we can instruct the AI to think in the way of a professional engineer to craft the most suitable and practical solutions. Without this approach, the answer we receive might come from a different perspective. While the summary of the log information would remain the same, the solution provided might not consider a site reliability engineer's viewpoint and could only offer solutions that a regular user of the website could implement. The 2nd point helps with clarification and ensure accuracy and acts as an extra precaution to ensure the most accurate response without missing out any detail. The 3rd and 4th instructions ensure the AI keeps the response concise and understandable, making the template suitable for users regardless of their prior experience with the system. The 5th point is also another protective measure to allow the response to be of optimal accuracy and ensure that the answer generated is holistic. Lastly, the 6th point helps keep everything under our budget to save cost as well as to keep the answer generated from elaborating unnecessarily.

In some cases, if there is no system message, the results from OpenAI may be extremely varied and not relevant to our request. OpenAI may give results that are out of topic and not useful for solving the problem at hand. For example, it may give us results where it just barely explains what the log details are instead of providing the troubleshooting methods in the correct order. This may cause unwanted confusion, and we have to repeatedly prompt OpenAI to give us the corrected response. Keeping prompts to the minimum would be beneficial as less tokens would be used up which will help save cost. Using a system method would be much more efficient as we also must factor in the time and cost as well as trying to produce effective results.

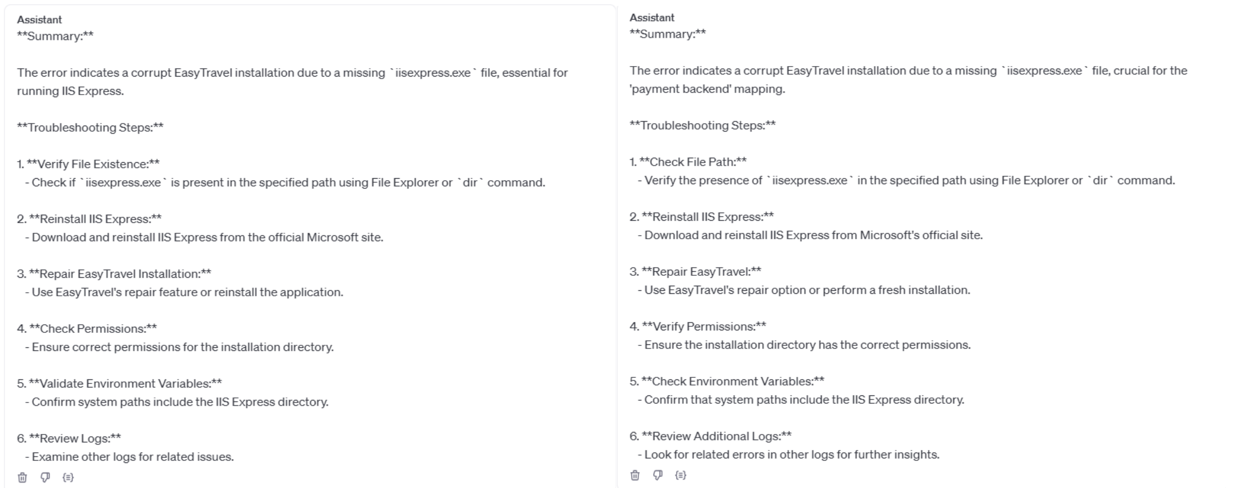
Finally, we can consolidate everything by inputting the specific error log message we are interested in. After entering all the necessary information, OpenAI will generate an immediate response that summarizes the error log message and provides straightforward solutions to resolve the errors. Below are the error logs used for this example.

2024-12-03 04:48:05 Launcher ERROR [ProcedureFactory] The easyTravel installation is corrupt. Unable to create procedure for mapping 'b2b frontend': Corrupt installation. The following file could not be found: C:\dev\..\..\IIS Express\iisexpress.exe.

2024-12-03 04:44:56 Launcher ERROR [ProcedureFactory] The easyTravel installation is corrupt. Unable to create procedure for mapping 'payment backend': Corrupt installation. The following file could not be found: C:\dev\..\..\IIS Express\iisexpress.exe.

The output is shown below:

OUTPUT



As demonstrated above, the AI can deliver a concise response, providing the essential information and summarizing it for the user. Furthermore, it provides us with a list of possible solutions to troubleshoot the error without overloading the engineer with too much information. The response generated is precise and able to correctly identifies the problem and recommend suitable solutions.

One important thing to take note of is the cost. Although the use of AI can significantly help with our task, it comes with a price while doing everything manually does not require any cost. Using OpenAI's token counter, the input is about 64 tokens with the output being 165 tokens. Using the numbers from above, we can calculate the estimated cost of one prompt being about $0.01182 which is about 1 cent. This is very favourable for any enterprise and is a good result for our research.

6.2 Problems faced and Limitations

At the beginning of the project, we were not very familiar with much of the terminology used in AI which made it difficult for us to understand the effect of changing the parameters in the OpenAI playground. Moreover, we could not interpret the logs correctly which caused us to struggle and understand the effect of shutting down the specific servers. Our inexperience caused us to struggle slightly but we were able to learn at our own pace and grasp the content. Moreover, due to our inexperience with programming, coming up with a way to filter the logs proved to be a difficult challenge, we used multiple online resources to gain more knowledge and work our way towards our finished product which took quite some effort as it was not working when we first started working on it.

6.3 Future use and Extension of our project

Some additional feature that may help with this program is to ask AI to help sort the error messages based on severity to help the engineers tackle the more serious errors first. Additionally, we can consider implementing AI agents to automate troubleshooting, further enhancing the system's efficiency. Furthermore, we realised through our research that the same error messages can come out repeatedly if the issue is not resolved. This would be a waste of resources if we were to repeat the prompt as we already have the troubleshooting recommendation and solutions from past error messages. Thus, one potential solution is to cache the past answers as reference and check each time before entering a log message into a prompt. If the log message matches a previous search, we can reference the past solution, avoiding the cost of calling the API for the same answer. Additionally, we can create a user-friendly and straightforward dashboard to consolidate errors and solutions, making it easily accessible to site reliability engineers for their assistance.

7. Conclusion

In conclusion, our extensive research demonstrated that we can effectively leverage the power of AI to alleviate the workload of Site Reliability Engineers (SREs). Their job of sifting through endless lines of logs is truly tiring and tedious and we experienced it ourselves in a demonstration of easyTravel. However, performing such tasks in a real-world context can be even more challenging, as they must be completed at a much faster pace with high accuracy, which can be stressful for Site Reliability Engineers. Therefore, with the introduction of our solution to make use of the various technologies available to us in today's world, we are able to reduce the stress of Site Reliability Engineers by automating such tedious tasks and helping them with their day-to-day work. While our solution may not yet be fully optimized and perfected, we believe we are on the right path toward a future where AI is used more efficiently to help not only Site Reliability Engineers but everyone to work smarter, not harder.

8. Acknowledgements

We would like to thank our mentors, Zhang Chenghai Christopher, Ng Rui Wei, Alvin, Lee Ming Liang and Kwok Jun Kiat for providing us with the support and guidance throughout our research project, constantly imparting new knowledge into us making our experience fruitful and enjoyable.

References

[1] Brockman, G., Sutskever, I., & OpenAI Team. (2015). Introducing OpenAI. Retrieved December 15, 2019, https://blog.openai.com/introducing-openai/(15.12.2019)

[2] Patil, N. (2023, June 13). Efficient Log Analysis with ChatGPT: Boosting QA Efficiency › TESTINGMIND. TESTINGMIND › Connecting QA Professionals Globally. https://www.testingmind.com/37045-2/

[3] What is Systems Engineering. (n.d.). INCOSE https://www.incose.org/about-systems-engineering/what-is-systems-engineering

[4] GeeksforGeeks. (2023, December 19). What is a System Engineer Skills, Roles and Responsibilities. GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/system-engineer/

[5] McGrath, A., & Jonker, A. (2023, December 2). What is model-based systems engineering (MBSE)? | IBM.https://www.ibm.com/topics/model-based-systems-engineering

[6] Chia, A. (2024). Log Analysis: A Complete Introduction | Splunk. Splunk; Splunk. https://www.splunk.com/en_us/blog/learn/log-analysis.html

[7] What is Log Analysis? | CrowdStrike. (2024). Crowdstrike.com. https://www.crowdstrike.com/en-us/cybersecurity-101/next-gen-siem/log-analysis/

[8] Bockes, B. (2022, December 21). The OpenAl Playground: A Beginner's Guide for Non-Developers | Medium. Medium; Medium. https://medium.com/@brendanbockes/the-openai-playground-a-beginners-guide-for-non-developers-a166dcb02094

[9] Mumuni, A., & Mumuni, F. (2024). Automated data processing and feature engineering for deep learning and big data applications: a survey. Journal of Information and Intelligence. https://doi.org/10.1016/j.jiixd.2024.01.002

[10] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June 10). Generative Adversarial Networks. ArXiv.org. https://arxiv.org/abs/1406.2661

[11] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[12] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models. ArXiv:2001.08361 [Cs, Stat]. https://arxiv.org/abs/2001.08361

[13] easyTravel Documentation and Download. (2022, February 23). Dynatrace.com. https://community.dynatrace.com/t5/Start-with-Dynatrace/easyTravel-Documentation-and-Download/m-p/181271

[14] Cleary, D. (2024). Understanding OpenAI Parameters: Optimize your Prompts for Better Outputs. Prompthub.us; PromptHub. https://www.prompthub.us/blog/understanding-openai-parameters-how-to-optimize-your-prompts-for-better-outputs