International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 6 Issue 3 PP 136-145, 2025
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V6I3P117

Original Article

AI-Driven Log Summarization for Security Operations Centers: A Web-Based Approach Using Gemini API

Raju Katukam
Site Reliability Engineer, Jawaharlal Nehru Technological University Hyderabad.

Received On: 15/07/2025 ***Revised On: 03/08/2025*** ***Accepted On: 05/09/2025*** ***Published On: 18/09/2025***

Abstract *- In today's cybersecurity landscape, Security Operations Centers (SOCs) face the growing challenge of managing and interpreting vast volumes of unstructured log data generated from diverse sources. Traditional rule-based monitoring approaches are often inefficient and inadequate in handling this scale, leading to delays in threat detection and response. This work presents a scalable, AI-powered log summarization platform that integrates Google's Gemini 1.5 Flash model within a full-stack web application to automate the extraction of security-relevant insights from log files. Built using React.js, Node.js, and MongoDB, the system enables SOC analysts to upload log files in various formats (.log, .txt, .docx) and receive real-time, human-readable summaries highlighting anomalies such as brute-force attacks, suspicious IP activity, and data exfiltration attempts. The backend handles secure file parsing, AI prompt generation, summarization via Gemini API, and summary storage with associated metadata. Experimental evaluations demonstrate low-latency summarization across file types, with average response times under four seconds, validating the platform's efficiency and practicality for real-world SOC environments. This solution significantly reduces manual analysis effort, enhances threat visibility, and introduces a flexible, extensible framework for AI-enhanced cybersecurity operations.*

Keywords *- Security Operations Center (SOC), Log Summarization, Generative AI, Gemini API, Node.js, React.js, Cybersecurity.*

1. Introduction

Large volumes of log data are generated by software systems as the program runs. These logs are often generated to assist system administrators and developers in identifying unusual output, runtime issues, and faults. Additionally, intrusion detection may make use of these logs. A system intrusion or assault may be detected and prevented by keeping tabs on critical events in the log data and identifying unusual behaviour or signatures[1]. It is not viable to manually analyse the logs since most software systems generate an excessive volume of log data[2][3][4]. Furthermore, dangers are often not detectable from single log entries but rather from a pattern of log entries dispersed across the file[5]. Therefore, it is essential to automate the process of analysing the log data. The majority of the focus in software cybersecurity is on safeguarding systems from potential outside attacks. The difficulty of detecting intrusions and internal threats causes these areas to be neglected[6][7][8]. Failure to take action against these dangers might result in significant financial losses if ignored. For real-time cybersecurity event detection, investigation, and mitigation, these logs are an essential source of information[9][10]. Cybersecurity attacks are becoming more frequent and complicated, therefore protecting digital assets successfully requires strong monitoring and quick reaction skills[11]. Therefore, SOCs are being set up by a growing number of organisations to actively identify and react to cybersecurity events[12][13].

SOCs play a pivotal role in continuously monitoring log data to ensure the resilience and security of organizational assets in cyber incidents[14][15][16]. However, as log data continues to grow exponentially in size and diversity, traditional manual and rule-based monitoring techniques are increasingly insufficient, necessitating more intelligent approaches to support efficient analysis and decision-making[17][18]. This growing demand for efficiency has drawn significant attention to the role of advanced computational methods, particularly Artificial Intelligence (AI), in augmenting SOC operations[19]. In particular, advancements in Natural Language Processing (NLP) have unlocked new possibilities for automating the interpretation and summarization of unstructured log data. AI-driven summarization techniques, powered by Large Language Models (LLMs), can distill vast and complex log streams into concise, human-readable narratives[20], thereby improving situational awareness and accelerating incident response for SOC analysts. Among the latest developments, generative LLMs[21][22] such as Google's Gemini API demonstrate exceptional capabilities in understanding context, capturing subtle patterns, and producing coherent summaries from raw data. Harnessing these capabilities offers a promising direction toward building systems that embed AI-powered summarization directly into SOC workflows[23][24].

To enable practical and scalable deployment of such intelligent capabilities, this paper proposes a web-based approach that integrates [25]. AI log summarization within a user-friendly, real-time interface accessible from anywhere[26]. The system is designed to receive logs from various sources, process them using a summarization model, and present the distilled outputs through an interactive dashboard. This cloud-accessible solution ensures ease of deployment, centralized control, and seamless collaboration among security teams, regardless of geographic distribution.

A web-based platform that integrates the Gemini API for log summarization emerges as a particularly effective approach to operationalize these AI capabilities[27][28]. Web-based architectures provide accessibility, scalability, and seamless integration with existing SOC dashboards and tools, while the Gemini API ensures advanced language understanding tailored to the nature of security logs[29][30][31]. By combining these elements, it is possible to deliver actionable insights from raw logs through an intuitive interface, enabling SOC analysts to make informed decisions quickly and effectively. This integrated approach forms the basis of the present work, which focuses on designing and demonstrating such a solution in a practical setting. Leveraging generative models like Google's Gemini API within a web-based platform provides an innovative and scalable approach to real-time log analysis. This not only reduces the cognitive load on SOC analysts but also accelerates threat detection and decision-making, ultimately strengthening organizational cybersecurity posture.

1.1. Motivation and Contribution of Study

Security Operations Centers (SOCs) are today's primary defense against cyber threats for digital infrastructures. In an era of increasing size and complexity, the number of logs produced every day by endpoints, servers, firewalls, applications, etc. continues to swell. Manual review of these logs takes time-and many times results in the late discovery of a threat or missing a key piece of information. Furthermore, most tools today rely on static rules-based or keyword-based; neither are effective for recognizing attack patterns or evolving threat landscapes. The emergence of LLM models that can read and interpret unstructured data, provides SOCs with a real opportunity to transform their operations by using natural language summarization in conjunction with log data. This study presents a novel AI-driven log summarization platform specifically designed for SOC environments. The key contributions are as follows:

- This work presents the development of a modular, full-stack web-based platform tailored for Security Operations Centers (SOCs), combining React.js for the frontend, Node.js/Express.js for the backend, and MongoDB for persistent storage. The architecture supports secure log file upload, real-time AI summarization, and efficient result visualization.
- The system is capable of processing .log, .txt, and .docx files using file-type-specific parsers (Node's fs module and the Mammoth library). This multi-format compatibility ensures wide applicability across varied SOC log sources.
- The platform incorporates Gemini's advanced language model via Google's Generative AI SDK to automatically generate concise, human-readable summaries of unstructured log content, extracting insights such as suspicious IPs, brute-force attacks, and data exfiltration attempts.
- Custom prompt templates are designed to guide the Gemini model toward detecting cybersecurity-relevant patterns, improving the contextual precision of AI-generated summaries without requiring predefined rules or manual annotations.
- Through empirical evaluation, the system demonstrated an average response time of under four seconds across various file types, validating its practicality for real-time threat triage and analysis in fast-paced SOC workflows.
- Security best practices such as JWT-based authentication, automatic post-processing file deletion, and metadata-enriched summary storage in MongoDB are implemented to ensure data integrity, user accountability, and compliance with audit requirements.

These contributions aim to empower SOC teams with a scalable and intelligent solution that reduces analyst fatigue, accelerates incident response, and modernizes log handling through AI-driven automation.

1.2. Novelty of the Study

This study introduces a new paradigm through the combination of Google's Gemini 1.5 Flash large language model and a secure, online Web application for logging summarization that can be done in real-time at Security Operations Centers (SOCs). Unlike traditional A/V log workflows which exist in a rule-based ecosystem, the system developed uses generative AI to convert raw, unstructured log data (.log, .txt, .docx) to compressed, meaningful summaries without human intervention or required parsing rules. The application encompasses a React.js front end and Express.js back end with MongoDB for storage, enabling a seamless end-to-end workflow from upload to AI-based output. The platform provides secure shared access to SOC personnel using a secure JWT authentication system and also performs automatic cleanup of processed documents to keep each SOC's operational footprint and data private and succinct. The framework for combining large language model competency with meaningful SOC-based operational use is unique, allowing analysts to process logs more quickly with reduced manual input while promoting improved incident response.

1.3. Structure of the paper

This is how this paper is organised: Section II examines related research in log analysis and summarisation using AI. Section III discusses the methodology for system design and integration of the Gemini API within a web-based interface. Section IV and V presents implementation analysis and experimental results with performance evaluations on real-world log datasets. Key results, limits, and future research directions are finally discussed in Section VI.

2. Literature Review

This section reviews recent work on AI-driven log analysis and summarization in Security Operations Centers (SOCs). Existing studies focus on enhancing threat detection and incident response using AI/ML, but few address real-time log summarization. Some existing studies for this work related discussed below:

Mohsin et al. (2025), article lays out a methodical plan for how humans and AI might operate together in SOCs, including features like trust calibration, human input during decision-making, and AI autonomy. Human supervision, trust calibration, and scalable autonomy with AI are largely neglected in existing SOC systems, which primarily concentrate on automation. Many people overlook the fact that SOC activities involving human and AI cooperation might vary greatly in complexity, criticality, and danger because they presume static or binary autonomy settings. To overcome these constraints, we suggest a new autonomy tiered structure based on five AI autonomy levels, ranging from totally autonomous to manual, that are correlated with task-specific trust thresholds and Human-in-the-Loop (HITL) responsibilities. This allows for the integration of adaptive and explicable AI into key SOC operations, such as incident response, threat detection, alert triage, monitoring, and protection[32].

Ismail et al. (2025), the need to update Security Operations Centres (SOCs) to better identify, respond to, and mitigate cybersecurity attacks is growing in response to the dynamic nature of these threats. Traditional no-code SOAR solutions have considerable limitations, such as limited flexibility, scalability issues, insufficient support for advanced logic, and difficulties in managing large playbooks. Despite their importance, security orchestration, automation, and response (SOAR) platforms are not without their flaws. There has to be a more complex solution since these limitations prevent analysts' technical competence from being fully used, limit flexibility, and impede successful automation. They suggest an agentic-LLM-powered hyper-automation SOAR platform to optimise automation processes by using Large Language Models (LLMs). By replacing inflexible no-code playbooks with AI-generated code, this method streamlines operations while increasing flexibility and scalability[33].

Park et al. (2025), paper presents an ATIRS, a SLM-based framework designed to automate the summarization of Suricata network alert logs and generate response recommendations for maritime environments. ATIRS enables actionable countermeasures like IP blocking or account lockout and transforms unstructured alarms into organised summaries. It becomes better with time by responding to user input and new threats via adaptive learning. Results from shipboard data show a significant reduction in Mean Time to Respond (MTTR), supporting efficient threat mitigation by non-expert crew in resource-limited settings[34]. Afridi and Abbas (2024), rapidly evolving cyber threat landscape, SOCs must adopt innovative technologies to enhance their efficiency and effectiveness. AI and ML are transforming SOC operations by automating threat detection, response, and mitigation processes. SOCs are able to analyse massive volumes of security data in real-time, spot irregularities, and anticipate cyber-attacks due to these cutting-edge technology. AI-driven SOCs leverage machine learning algorithms to continuously improve threat detection by identifying patterns and behaviors indicative of malicious activities. This reduces reliance on manual intervention, minimizes false positives, and accelerates incident response times. AI-driven behavioral analytics enable SOCs to detect zero-day attacks and insider threats with greater accuracy [35].

Balaji et al. (2024), paper reviews advanced log anomaly detection techniques that employ AI technologies, with a specific focus on the Isolation Forest Algorithm. We delve into the integration of Endpoint Detection and Response (EDR) tools, pivoting techniques, process tree analysis, and summarization methods to enhance the identification and interpretation of suspicious activities. The paper discusses the development and examination of process trees, aimi ng to equip SOC analysts with practical insights and recommendations. Additionally, we assess how AI-powered log analysis can overcome existing challenges and improve the detection of complex threats. Finally, we summarize our key findings and propose future research avenues to address ongoing challenges in log anomaly detection[36]. Balasubramanian et al. (2024), a GPT-3.5 Turbo-powered conversational agent framework, CYGENT, is developed to support system administrators in addressing cybersecurity challenges across IT and IoT environments. CYGENT performs tasks such as log file summarization, event detection, and user guidance. Fine-tuning of GPT-3 models with custom data achieved a BERTscore above 97%, demonstrating strong summarization performance. Comparative analysis with other Large Language Models, including CodeT5-small, CodeT5-base, and CodeT5-base-multi-sum, shows that the GPT-3 (Davinci) model consistently outperforms others, while CodeT5-base-multi-sum demonstrates potential as an effective offline alternative. These findings highlight the effectiveness of generative AI in enhancing cybersecurity operations and log analysis[37].

Mudgal et al. (2023), paper investigates the capabilities of ChatGPT in processing complex log data from large-scale software systems, a topic that has received limited attention. While ChatGPT has shown promise in tasks like code generation and summarization, its performance in log analysis remains limited, showing inconsistencies and scalability issues. The study identifies key shortcomings and outlines potential directions to enhance the effectiveness of large language models in log processing, aiming to support future research in this area[38]. Recent work on AI-driven log analysis in Security Operations Centers (SOCs) has focused on enhancing threat detection, automation, and Human-AI collaboration. Studies have introduced structured frameworks for integrating AI autonomy with human oversight, improving decision-making in core SOC functions. Hyper-automation using large language models (LLMs) has been explored to overcome limitations in traditional SOAR platforms, offering flexible and scalable alternatives to no-code solutions. Domain-specific models have shown success in summarizing network alerts and recommending responses, particularly in constrained environments like maritime systems. AI and ML techniques are increasingly used for anomaly detection, behavioral analytics, and threat prediction, while conversational agents powered by models like GPT-3.5 have demonstrated strong summarization capabilities.

However, major research gaps persist, including the limited focus on real-time log summarization, poor generalizability across domains, lack of explainability for non-expert users, scalability issues in LLMs, underutilization of lightweight small language models (SLMs), and minimal integration of adaptive learning based on user feedback highlighting the need for more interpretable, efficient, and domain-flexible AI systems for SOC log analysis. To address these challenges, the proposed solution introduces a modular AI-powered log summarization system that leverages LLMs to generate real-time, context-aware summaries. The system incorporates secure data handling, adaptive learning, and a user-friendly interface to support SOC analysts in efficiently managing and interpreting large-scale log data.

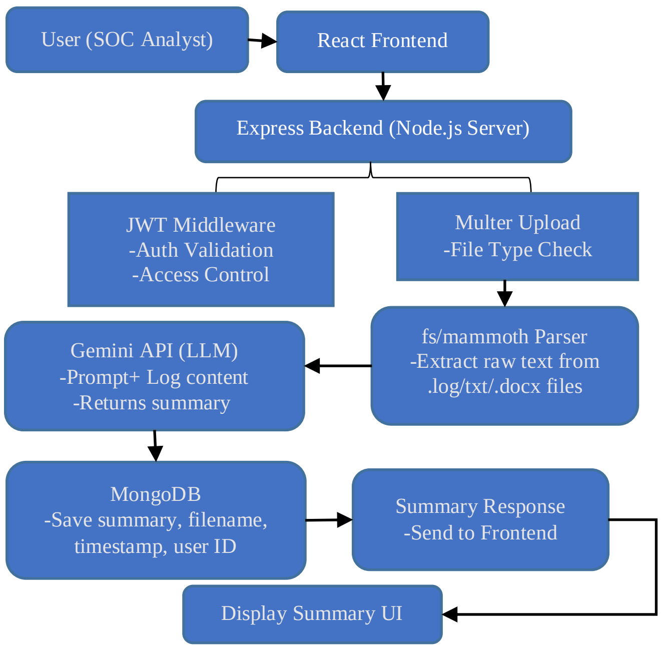

User (SOC Analyst)
React Frontend
Express Backend (Node.js Server)
JWT Middleware -Auth Validation -Access Control
Multer Upload -File Type Check
fs/mammoth Parser -Extract raw text from .log/txt/.docx files
Gemini API (LLM) -Prompt+ Log content -Returns summary
MongoDB -Save summary, filename, timestamp, user ID
Summary Response -Send to Frontend
Display Summary UI

Fig 1: System architecture of AI-driven SOC log summarization using Gemini API

3. Methodology

The aim of this work is to develop a scalable, AI-powered log summarization platform for Security Operations Centers (SOCs) that reduces manual analysis time and enhances incident response. The system architecture, shown in Figure 1, comprises a web-based interface built with React.js, enabling analysts to securely upload log files and receive AI-generated summaries. Uploaded files are transmitted via secure HTTP requests to a Node.js/Express backend, where file validation and text extraction are handled using Multer middleware, the fs module, or Mammoth (for .docx files). Extracted text is embedded into a custom prompt and sent to Google's Gemini 1.5 Flash model via the Generative AI SDK. The returned summary highlights key insights such as suspicious IPs, malware signs, or brute-force attempts. Summaries are rendered on the frontend and stored in MongoDB with metadata, while the original files are deleted for security. This modular design supports future extensions and enables SOCs to integrate AI seamlessly into their log analysis workflow.

Here is a more detailed explanation of the technologies used in the application:

3.1. Frontend (Client-Side)

The client interface is developed using React.js, supported by modern JavaScript and CSS for a responsive and intuitive experience. It provides analysts with secure file upload functionality, real-time AI summary visualization, and token-based authentication using JWT.

- **React.js:** The reason for React's success is its numerous notable features. It supports JSX, is component orientated, and is easy to use, among other things. Another component of React is the virtual DOM, which is responsible for the enhanced UI performance[39].A component-based framework used to manage dynamic UI updates, file upload events, and API responses . Hooks such as useState and useEffect are used to manage user interactions and render summaries returned by the backend.
- **Security:** JWT tokens are stored securely and attached to HTTP headers to protect summary requests.

3.2. Backend (Server-Side)

The backend is implemented using Node.js and Express.js, acting as the middleware layer that manages file processing, AI integration, user authentication, and database interactions. It coordinates the full summarization workflow from receiving log files to generating summaries using the Gemini 1.5 Flash model.

- **Node.js:** The Node.js framework is based on the Chrome V8 JavaScript engine, which is a high-performance compiler and interpreter that runs JavaScript programs at about the speed of light. It is well-suited for developing real-time web applications since this framework allows Node.Js to effectively handle many concurrent connections [40]. Provides a high-performance, asynchronous runtime environment that efficiently handles concurrent file uploads, log parsing, and communication with external APIs such as Gemini.

- **Express.js:** Node.js (JavaScript Runtime Environment) is the foundation of Express.js, a web app framework that is well-known for its flexibility and simple design, making it one of the most popular Node.js frameworks [41]. Acts as the routing layer, exposing endpoints such as /api/upload, /api/summary, and /api/login. It handles file validation, user session management, and Gemini API integration.
- **File Processing and Auto Cleanup:** Uploaded files are processed using Multer, which validates the file type and stores them temporarily in the /uploads directory. The backend then extracts content using either Node.js's built-in fs module for .log and .txt files, or the mammoth library for .docx files to ensure clean and formatting-free text extraction. After the Gemini API generates the AI summary, the system performs an automatic cleanup using fs.unlink() to delete the uploaded file from the server.
- **JSON Web Token (JWT):** Lightweight is a word that best describes JSON web tokens. Using these tokens, there is minimum database interaction required for client-server data exchange, user authentication, and authorisation. When it comes to protecting sensitive information from potential attackers, JWT employs a number of cryptographic encryption methods [42]. The backend uses JWT for stateless and secure authentication. For each request that our service receives, if the user is authenticated, there is a token in the HTTP headers. If it is present, we will validate it before providing access to these protected resource routes. This means only properly authorized users will be able to upload files, get summaries, and interact with the service.

3.3. Gemini API Integration

In this work, Google's Generative AI (Gemini API) is integrated into a full-stack web application designed for Security Operations Centers (SOCs) to process uploaded log files and generate concise, context-aware summaries. The integration enhances the system's ability to analyze unstructured log data in real time through advanced natural language understanding and prompt engineering. Gemini API enables accurate detection of anomalies, threats, and unusual activity patterns, providing actionable insights to analysts. At the core of the system is the integration with Google's Gemini 1.5 Flash model via the official Generative AI SDK.

3.4. Database (MongoDB)

An open-source document database, MongoDB offers autonomous scalability, high availability, and great performance. In MongoDB, a document containing field and value pairs is considered a record. You may think of MongoDB documents as JSON objects [43].The system uses MongoDB as the backend database to store AI-generated summaries, along with metadata such as filename, timestamp, and user ID. Using Mongoose, schemas are enforced for consistency, and summaries can be retrieved for audit or future analysis. This storage layer enables secure logging, summary history, and potential integration with other SOC monitoring tools.

4. Implementation and Technical Setup

This section details the end-to-end configuration, tools, and design choices implemented to develop the proposed AI-driven log summarization platform for Security Operations Centers (SOCs). The system is built using modern web technologies and integrates Google's Gemini 1.5 Flash model to generate intelligent summaries of uploaded security log files. The architecture follows a modular and secure design, enabling SOC analysts to perform fast, reliable, and scalable log summarization directly from a web interface.

4.1. System Overview

The system is built as a full-stack web application built from a React.js front-end, Node.js/Express.js back-end and a MongoDB database. It enables users to upload log files in common formats (.log, .txt, .docx) and receive AI summaries produced by the Gemini 1.5 Flash API. The application uses JSON Web Tokens (JWT) for secure access to the profiles and it has implemented automatic file parsing, summarization and saving. The back-end uses the official SDK to interact with Google's Generative AI service to properly parse and summarize the unstructured log content into significant actionable security insights.

4.2. Backend Implementation.

The AI-log summarization system's backend is built with Express.js, which is the main orchestrator that connects the React frontend, the file system, the Gemini API, and the MongoDB. When a file is uploaded, the backend uses Multer middleware to validate that the upload is a file and extracts the raw text using the Mammoth library for .docx files, or the Node.js fs module for .log/.txt files. The extracted raw text is placed into a structured prompt that instructs the Gemini 1.5 Flash model to assess and extract threats or anomalies. The API response returns the results from the model in a summary ready for a human to read. The model summary is saved to MongoDB along with metadata (filename, timestamp, user ID). The backend uses fs to delete the status on the server file system and returns the model summary to the frontend for review by an analyst.

Listing pseudocode 1: Backend File Processing

```
Function handleFileUpload(request):
   Validate file type (.log, .txt, .docx)
   If .docx:
      Extract content using mammoth parser
   Else:
      Read content using fs module
   Generate prompt = "Summarize the following logs:\n\n" +
content
   summary = call Gemini API(prompt)
   Store summary, filename, user ID, timestamp in MongoDB
```

This pseudocode 1 delivers a straightforward high-level representation of the critical logic behind the backend functionality. The modular flow ensures that each aspect of the backend operation from validation to cleanup is carried out quickly and securely. By forming this flow, it is possible to drive rapid processes, provide secure access for sensitive log data, and smooth integration with the AI summarization web service. In addition, the flow allows for scalable and extensible processing to allow later options for additional file formats or to create rules based on customized processing.

4.3. Gemini API Integration

The AI based summarization approach is largely consisted of the integration with Google's Gemini 1.5 Flash model. This model performs an automated and natural language real time summarization of unstructured log data. This is a key area of the system, as the AI model is capable of synthesizing large log entries into many shorter summaries focused on key security events. Once an unstructured log is parse by the backend, a prompt for the domain is designed and sent to the Gemini model via RESTful API call aided by Google's Generative AI SDK. The prompt template is crafted to instruct Gemini to detect cybersecurity-relevant details such as suspicious IP addresses, brute force attempts, access anomalies, or malware signatures.

- **Prompt Template Example:** "Summarize the following SOC log entries. Identify any security risks, threats, or anomalies:\n\n[log_content]"

The following pseudocode 2 outlines the logic used by the backend to interact with the Gemini API and retrieve the summarization output:

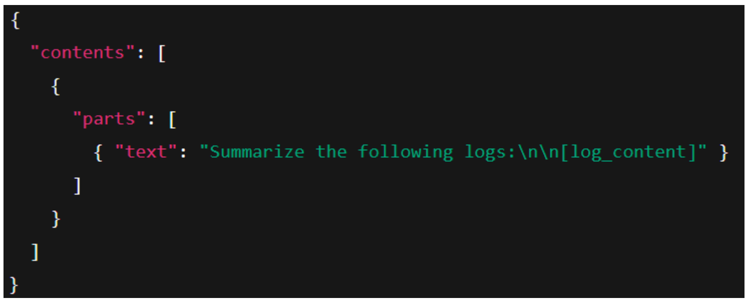
Listing pseudocode 2: Gemini API Invocation Logic

```
Function callGeminiAPI(prompt):
  Set endpoint =
"https://generativelanguage.googleapis.com/v1beta/models/gemin
i-1.5-flash:generateContent"
  Set headers including Content-Type and API key from
environment variables
  Format request body with a 'contents' array containing the
prompt text

  Send POST request to Gemini endpoint with headers and body

  If response is successful:
    Extract summary from response object
  Else:
    Log error and return a default fallback message
  Return summary
```

Once the POST request is submitted, the Gemini API processes the prompt and returns a structured, human-readable summary. Below Figure 2 is an example of the JSON payload sent to the API:



```
{
  "contents": [
    {
      "parts": [
        { "text": "Summarize the following logs:\n\n[log_content]" }
      ]
    }
  ]
}
```

Fig 2: Example JSON payload

The responses generated by Gemini are subsequently parsed by the backend, and the resulting summary is then saved to MongoDB along with metadata such as the filename, user ID, and timestamp. Reasoning based intelligent summarization circumvent the need for rule-based parsers, works with various log formats, and provides scalable AI assistance to SOC analysts without manual customizations or pre-defined rules. By utilizing Gemini's ability in the SOC workflow it could enhance threat triage, reduce analyst burnout, and hasten the incident response by integrating AI-based summarization.

4.4. Frontend Implementation

The front end of the AI-based log summarization platform is built using 'React.js,' which offers a dynamic and responsive user interface for SOC analysts. The interface mainly allows users to upload log files securely, place summarization requests, and display the generated summaries from the AI in a user-friendly and organized manner. The front end uses Axios for HTTP communication with the backend API and Tailwind CSS for responsive style and supported for all devices. The main focus of the design was on usability, so SOC personnel can easily initiate use of the system without the need for technical training.

Listing pseudocode 3: File Upload and Summary Rendering

```
Function handleFileUpload(file):
  Use FileReader to read content
  Store text in React state

Function handleSummarize():
  Send POST request with file content and JWT
  Receive summary from backend
  Display summary to user
```

The frontend is developed in React.js and allows authenticated users to upload .log, .txt or .docx files which are read with the FileReader API. handleFileUpload(file), as shown in Pseudocode 3, reads the content into React state; then, handleSummarize() sends to the backend the content as well as the JWT token, using Axios. The backend sends the AI generated summary which the frontend displays in a scrollable component. This streamlined interface supports efficient interaction as well as will support future enhancements, such as filtering, and tagging.

4.5. MongoDB and Security Setup

The system uses MongoDB to store AI-generated summaries along with metadata like filename, timestamp, and user ID. After summarization, the backend creates a structured object and inserts it into the database using Mongoose. JWT authentication secures all API routes, ensuring only authorized users can access or submit data. Environment variables are used to protect credentials and API keys.

Listing pseudocode 4: MongoDB Storage Logic

```
Function saveSummaryToDB(summary, filename, userID):
    Create object with fields:
        - summary
        - filename
        - timestamp (current time)
        - userID (from authenticated session)

    Connect to MongoDB
    Insert object into the 'summaries' collection
    Close connection or await future access
```

As shown in Pseudocode 4, the backend creates an object containing the summary and related metadata and inserts it into the MongoDB collection. This approach ensures secure, user-specific storage of summarization results, enabling auditability and future retrieval.

5. Results and Discussion

This section evaluates the functionality and performance of the proposed AI-driven SOC dashboard, which leverages Google’s Gemini 1.5 Flash model for log summarization. The system was tested across multiple log formats and scenarios relevant to real-world SOC operations. Metrics such as file processing time, API latency, AI response quality, and user experience were captured to determine effectiveness. Additionally, sample summaries were analysed to demonstrate the contextual awareness of the LLM when analysing unstructured security log data.

5.1. Evaluation Setup

The system was deployed on a local development server using a Dell laptop equipped with an Intel Core i5 processor, 8 GB RAM, and a 256 GB SSD. The Gemini API was accessed over a stable internet connection averaging 50 Mbps. Logs of various formats, including .log, .txt, and .docx, were uploaded for testing. The backend was implemented in Node.js and Express.js, while the frontend utilized React.js. MongoDB was used for storing user sessions and summary metadata.

5.2. Performance Metrics

To quantify system performance, multiple trials were conducted for each file type. The following metrics were recorded over 10 iterations:

- **Upload to Summary Time:** Time taken from file upload to the display of AI-generated summary.
- **Gemini API Response Time:** Time taken by the Gemini API to return a summary after receiving the prompt.
- **MongoDB Write Latency:** Time taken to store the summary and metadata in the database.

Table 1: Performance Metrics by File Type

Metric	.log	.txt	.docx
Upload to Summary Time (sec)	2.1	1.8	3.4
Gemini API Response (avg)	1.2	1.1	2.0
MongoDB Write Latency (ms)	38	34	40

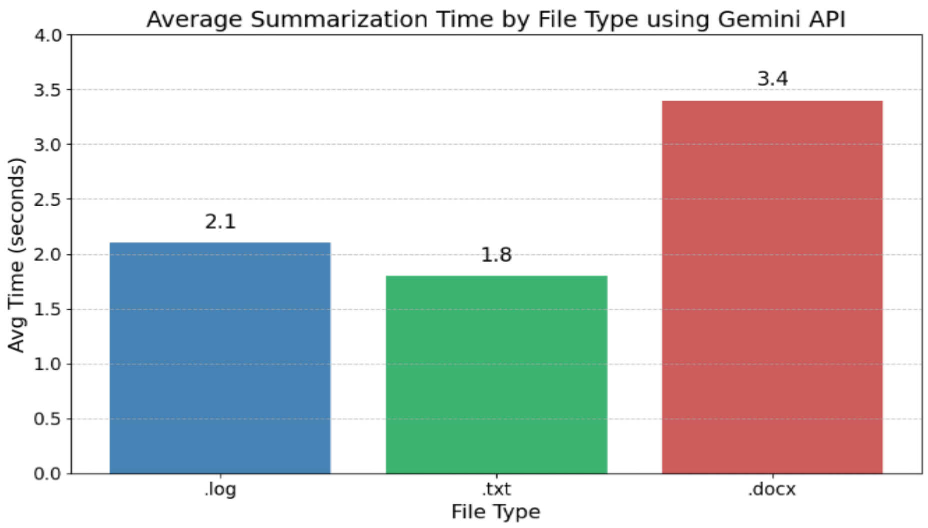


Fig 3: Average summarization time by file type using Gemini API.

Table I and Figure 3 present performance evaluation measures the system's response to a variety of log file formats. The various metrics calculated include upload-to-summary time, Gemini API response time, and MongoDB write latency. Based on the evaluation results it was found that .txt and .log files experienced a faster processing time and an average summary time of 3 seconds or less, and .docx files experienced an average summary time of 4 seconds, which can be attributed to slightly more parsing overhead. Figure 3 visually compares these average times, confirming the system's ability to deliver real-time summarization performance across formats, making it efficient and suitable for SOC environments.

5.3. Sample Output Summaries

These examples demonstrate how the Gemini API interprets and summarizes diverse logs from .log, .txt, and .docx files helping SOC analysts quickly detect anomalies and security events.

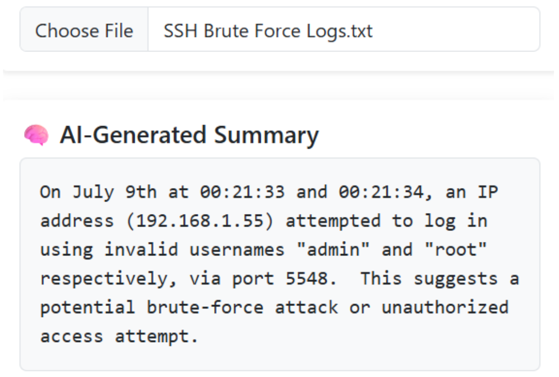
5.3.1. Sample 1: SSH Brute Force Logs

SSH Brute Force Logs.txt

Raw Log:

```
Jul 09 00:21:33 Failed password for invalid user admin from 192.168.1.55 port 5548
Jul 09 00:21:34 Failed password for invalid user root from 192.168.1.55 port 5548
```

Gemini Summary Output:



```
On July 9th at 00:21:33 and 00:21:34, an IP
address (192.168.1.55) attempted to log in
using invalid usernames "admin" and "root"
respectively, via port 5548.  This suggests a
potential brute-force attack or unauthorized
access attempt.
```

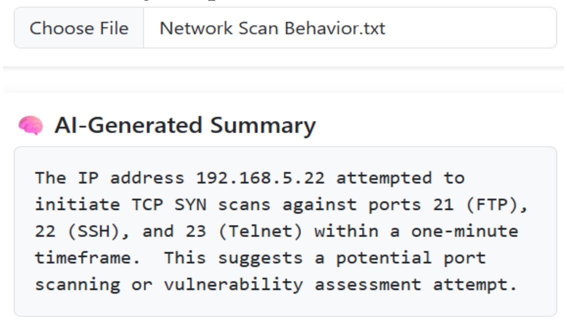
Fig 4: AI-Generated Summary brute-force attack

5.3.2. Sample 2: Network Scan Behavior

Input Log (.log):

```
[18:42:01] Incoming TCP SYN request from 192.168.5.22 to port 21 .
[18:42:02] Incoming TCP SYN request from 192.168.5.22 to port 22  .
[18:42:03] Incoming TCP SYN request from 192.168.5.22 to port 23.
```

Gemini Summary Output:



```
The IP address 192.168.5.22 attempted to
initiate TCP SYN scans against ports 21 (FTP),
22 (SSH), and 23 (Telnet) within a one-minute
timeframe.  This suggests a potential port
scanning or vulnerability assessment attempt.
```

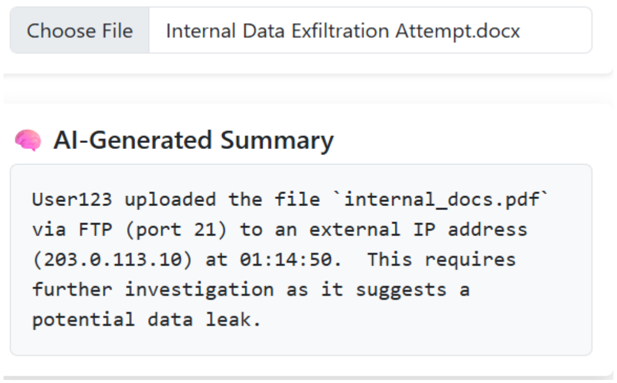
Fig 5: AI-Generated Summary brute-force attack

5.3.3. Sample 2: Internal Data Exfiltration Attempt

Input Log (.txt):

```
[01:14:50] File upload: internal_docs.pdf by user123
[01:14:52] Destination IP: 203.0.113.10
[01:14:55] Protocol: FTP, Port: 21
```

Gemini Summary Output:



```
User123 uploaded the file `internal_docs.pdf`
via FTP (port 21) to an external IP address
(203.0.113.10) at 01:14:50.  This requires
further investigation as it suggests a
potential data leak.
```

Fig 6: AI-Generated Summary of Internal Data Exfiltration Attempt

Figures 4 to 6 illustrate the effectiveness of the AI-driven summarization system in identifying and interpreting different cybersecurity events from raw log inputs. Figure 4 captures a classic SSH brute-force attack, where repeated failed login attempts from the same IP indicate unauthorized access attempts. Figure 5 highlights network scanning behavior, with sequential SYN requests to multiple ports suggesting reconnaissance activity from a potential attacker. Figure 6 demonstrates the detection of an internal data exfiltration attempt, where a sensitive file is uploaded via FTP to an external IP address. These examples show how the Gemini API successfully transforms unstructured logs into concise, contextual summaries, enabling SOC analysts to quickly recognize and prioritize security threats.

5.4. Observations and Discussion

Throughout implementation and evaluation there were a key observation regarding the operational performance few, functional utility, and usability of the system within a SOC. First, the integration of the Gemini 1.5 Flash model successfully provided contextually aware and accurate summarizations of log files without requiring users to build rules or patterns. Clearly, the model was able to identify security events and trends which are crucial in security investigations: bruteforce attempts; suspicious IP addresses; and patterns of suspected data exfiltration; among various distinct types of log-files (.log, .txt, and .docx). From a performance testing perspective, there was minimal latency in response timing for summarization and total time to process logs was, on average, under 4 seconds per log. Time to process the logs of the .docx document were slightly slower than for the other log types but there was still no major issue for operational usability. Given MongoDB write latency was minimal and able to confirm the feasibility of real-time storage and retrieval aspects of the application. Overall, the system demonstrated high usability, strong AI-model integration, and operational efficiency, validating its suitability for modern SOC workflows and establishing a foundation for further enhancements such as real-time stream processing or multi-log correlation.

5.5. Justification and Advantages

The proposed AI-driven log summarization platform is justified by the growing need for efficient and scalable solutions in Security Operations Centers (SOCs), where traditional manual or rule-based log analysis methods often fall short due to the volume and complexity of security data. By integrating Google's Gemini 1.5 Flash model, the system provides real-time, context-aware summaries of unstructured log files without requiring predefined rules or templates. Its full-stack web-based architecture offers secure, user-friendly access via JWT authentication, allowing analysts to process .log, .txt, and .docx files quickly and remotely. The main advantages include reduced analyst workload, faster incident response, format flexibility, and seamless integration with existing SOC tools. This intelligent and modular solution modernizes log analysis and enhances the overall responsiveness and effectiveness of SOC operations.

6. Conclusion and Future Scope

This paper introduced an AI-driven log summarization platform designed specifically for Security Operations Centers (SOCs), integrating Google's Gemini 1.5 Flash model within a secure, full-stack web architecture. The system automates the conversion of unstructured log data into concise, actionable summaries, enabling SOC analysts to detect security anomalies such as brute-force attacks, scanning behavior, and data exfiltration attempts. Built using React.js for the frontend, Node.js/Express for the backend, and MongoDB for persistent storage, the platform supports multiple log file formats (.log, .txt, .docx), employs JWT-based authentication, and ensures secure data handling through automated file deletion. Performance evaluation showed that the platform consistently delivered summaries with end-to-end latency under four seconds, demonstrating its suitability for real-time SOC environments and operational scalability.

Despite its effectiveness, the platform has some limitations. Currently, it operates in a batch-processing mode and lacks real-time stream log ingestion, which restricts its application in continuous monitoring environments. The summarization accuracy is dependent on prompt design and may be affected when handling highly domain-specific or multilingual logs. Additionally, the absence of feedback loops limits the system's ability to learn from analyst corrections or improve over time. In the future, enhancements will focus on supporting live stream log ingestion, integration with SIEM platforms for real-time threat detection, and expanding summarization to include multiple languages. Further developments will also include adding explainable AI (XAI) components for transparency, feedback-driven refinement of summaries, and advanced classification modules to automatically tag and prioritize detected security threats.

References

[1] D. Vavpotič, S. Bala, J. Mendling, and T. Hovelja, "Software Process Evaluation from User Perceptions and Log Data," *J. Softw. Evol. Process*, vol. 34, no. 4, pp. 1–14, Apr. 2022, doi: 10.1002/smr.2438.

[2] A. Abhishek and P. Khare, "Cloud Security Challenges: Implementing Best Practices for Secure SaaS Application Development," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 06, Nov. 2021, doi: 10.14741/ijcet/v.11.6.11.

[3] Vikas Prajapati, "Role of Identity and Access Management in Zero Trust Architecture for Cloud Security: Challenges and Solutions," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 5, no. 3, pp. 6–18, Mar. 2025, doi: 10.48175/IJARSCT-23902.

[4] V. Thangaraju, "Security Considerations in Multi-Cloud Environments with Seamless Integration: A Review of Best Practices and Emerging Threats," *Trans. Eng. Comput. Sci.*, vol. 12, no. 2, pp. 1–6, 2024.

[5] K. S. Jeon, S. J. Park, S. H. Chun, and J. B. Kim, "A study on the big data log analysis for security," *Int. J. Secur. its Appl.*, 2016, doi: 10.14257/ijsia.2016.10.1.02.

[6] M. Wurzenberger, F. Skopik, R. Fiedler, and W. Kastner, "Discovering insider threats from log data with high-performance bioinformatics tools," in *MIST 2016 - Proceedings of the International Workshop on Managing Insider Security Threats, co-located with CCS 2016*, 2016. doi: 10.1145/2995959.2995973.

[7] V. S. Thokala, "Improving Data Security and Privacy in Web Applications : A Study of Serverless Architecture," *Int. Res. J.*, vol. 11, no. 12, pp. 74–82, 2024.

[8] S. P. Kalava, "Enhancing Software Development with AI-Driven Code Reviews," *North Am. J. Eng. Res.*, vol. 5, no. 2, pp. 1–7, 2024.

[9] D. Tovarňák, S. Špaček, and J. Vykopal, "Traffic and log data captured during a cyber defense exercise," *Data Br.*, 2020, doi: 10.1016/j.dib.2020.105784.

[10] V. Prajapati, "Cloud-Based Database Management : Architecture , Security , challenges and solutions," *J. Glob. Res. Electron. Commun.*, vol. 1, no. 1, 2025.

[11] Suhag Pandya, "Innovative blockchain solutions for enhanced security and verifiability of academic credentials," *Int. J. Sci. Res. Arch.*, vol. 6, no. 1, pp. 347–357, Jun. 2022, doi: 10.30574/ijsra.2022.6.1.0225.

[12] S. A. Chamkar, Y. Maleh, and N. Gherabi, "Security Operations Centers: Use Case Best Practices, Coverage, and Gap Analysis Based on MITRE Adversarial Tactics, Techniques, and Common Knowledge," *J. Cybersecurity Priv.*, vol. 4, no. 4, pp. 777–793, Sep. 2024, doi: 10.3390/jcp4040036.

[13] S. P. Godavari Modalavalasa, "Exploring Azure Security Center: A Review of Challenges and Opportunities in Cloud Security," *ESP J. Eng. Technol. Adv.*, vol. 2, no. 2, pp. 176–182, 2022, doi: 10.56472/25832646/JETA-V2I2P120.

[14] M. Vielberth, F. Bohm, I. Fichtinger, and G. Pernul, "Security Operations Center: A Systematic Study and Open Challenges," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.3045514.

[15] V. Panchal, "Mobile SoC Power Optimization : Redefining Performance with Machine Learning Techniques," *IJIRSET*, vol. 13, no. 12, pp. 1–17, 2024, doi: 10.15680/IJIRSET.2024.1312117.

[16] H. Mistry, K. Shukla, and N. Patel, "Transforming Incident Responses, Automating Security Measures, and Revolutionizing Defence Strategies through AI-Powered Cybersecurity," *J. Emerg. Technol. Innov. Res.*, vol. 11, no. 3, pp. h38–h45, 2024.

[17] R. de Céspedes and G. Dimitoglou, "Development of a Virtualized Security Operations Center," *J. Comput. Sci. Coll.*, 2021.

[18] V. S. Thokala, "A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications," *Int. J. Res. Anal. Rev.*, vol. 8, no. 4, pp. 383–389, 2021.

[19] S. B. Shah, "Machine Learning for Cyber Threat Detection and Prevention in Critical Infrastructure," *Dep. Oper. Bus. Anal. Inf. Syst. (OBAIS*, vol. 2, no. 2, pp. 1–7, 2025, doi: 10.5281/zenodo.14955016.

[20] Y. Lu, "LogSage: Log Summarization Assistant with Guided Enhancement," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, New York, NY, USA: Association for Computing Machinery, 2025, pp. 1979–1981.

[21] K. S. Saurabh Pahune, Zahid Akhtar, Venkatesh Mandapati, "The Importance of AI Data Governance in Large Language Models," *Preprints*, 2025.

[22] M. C. Saurabh Pahune, "Several categories of large language models (llms): A short survey," *arXiv*, 2023, doi: arXiv preprint arXiv:2307.10188.

[23] E. Ferragut and N. Braden, "System log summarization via semi-Markov models of inter-arrival times," in *ACM International Conference Proceeding Series*, 2011. doi: 10.1145/2179298.2179346.

[24] V. Thangaraju, "Enhancing Web Application Performance and Security Using AI-Driven Anomaly Detection and Optimization Techniques," *Int. Res. J. Innov. Eng. Technol.*, vol. 9, no. 3, p. 8, 2025.

[25] S. Locke, H. Li, T.-H. P. Chen, W. Shang, and W. Liu, "LogAssist: Assisting Log Analysis Through Log Summarization," *IEEE Trans. Softw. Eng.*, vol. 48, no. 9, pp. 3227–3241, 2022, doi: 10.1109/TSE.2021.3083715.

[26] S. Grizan and S. Gurun, "On-Device Log Summarization Using Artificial Intelligence to Improve Crash Analysis," 2024.

[27] B. P. Woolf, "Implementation Of A Gemini-Driven Adaptive Learning System For Personalized Online Education," vol. 13, no. 6, pp. 11–18, 2025.

[28] M. Menghnani, "Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications," *Int. J. Innov. Sci. Res. Technol.*, vol. 10, no. 2, pp. 1206–1216, 2025, doi: 10.5281/zenodo.14959407.

[29] R. Shrivastav, S. Shahane, T. S. Hydri, M. V Akre, and Z. D. Amin, "Exploring potential of Gemini with AI based content generator," *Int. J. Res. Comput. Inf. Technol.*, vol. 2, no. 1, pp. 68–72, 2024, doi: 10.5281/zenodo.11207604.

[30] S. P. B. and G. Modalavalasa, "Advancements in Cloud Computing for Scalable Web Development: Security Challenges and Performance Optimization," *J. Comput. Technol. Int. J.*, vol. 13, no. 12, pp. 01–07, 2024.

[31] S. Sesha and S. Neeli, "Data Protection in the Digital Age : SOC Audit Protocols and Encryption in Database Security," *ESP Int. J. Adv. Comput. Technol.*, vol. 2, no. 3, pp. 167–172, 2024, doi: 10.56472/25838628/IJACT-V2I3P115.

[32] A. Mohsin, H. Janicke, A. Ibrahim, I. Sarker, and S. Camtepe, "A Unified Framework for Human AI Collaboration in Security Operations Centers with Trusted Autonomy," 2025. doi: 10.48550/arXiv.2505.23397.

[33] Ismail *et al.*, "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence," *Information*, vol. 16, no. 5, 2025, doi: 10.3390/info16050365.

[34] D. Park, B. Min, S. Lim, and B. Kim, "ATIRS: Towards Adaptive Threat Analysis with Intelligent Log Summarization and Response Recommendation," *Electronics*, vol. 14, no. 7, p. 1289, Mar. 2025, doi: 10.3390/electronics14071289.

[35] S. Afridi and A. Abbas, "AI and Machine Learning-Driven SOC Operations: Transforming Cyber Security Efficiency," 2024. doi: 10.13140/RG.2.2.10444.53122.

[36] S. Balaji, D. Puspita, S. Sriram, and S. Ragul, "AI Enhanced Anomaly Detection of System Logs in Cyber Security," in *2024 International Conference on System, Computation, Automation and Networking (ICSCAN)*, IEEE, Dec. 2024, pp. 1–6. doi: 10.1109/ICSCAN62807.2024.10894286.

[37] P. Balasubramanian, J. Seby, and P. Kostakos, "CYGENT: A cybersecurity conversational agent with log summarization powered by GPT-3," in *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)*, IEEE, May 2024, pp. 1–6. doi: 10.1109/AIIoT58432.2024.10574658.

[38] P. Mudgal and R. Wouhaybi, "An Assessment of ChatGPT on Log Data," in *Communications in Computer and Information Science*, 2024. doi: 10.1007/978-981-99-7587-7_13.

[39] V. Komperla, P. Deenadhayalan, P. Ghuli, and R. Pattar, "React: A detailed survey," *Indones. J. Electr. Eng. Comput. Sci.*, 2022, doi: 10.11591/ijeecs.v26.i3.pp1710-1717.

[40] M. Kumawat, V. Shrivastava, A. Pandey, and S. Kumar, "International Journal of Research Publication and Reviews Node . Js Review : A Comprehensive Overview of the JavaScript Runtime Environment," *Int. J. Res. Publ. Rev.*, vol. 5, no. 4, pp. 268–270, 2024.

[41] A. Mardan, "Starting with Express.js," 2014, pp. 3–14. doi: 10.1007/978-1-4842-0037-7_1.

[42] A. Sharma, V. Shrivastava, A. Pandey, and E. A. Sharma, "International Journal of Research Publication and Reviews Providing Authentication using JSON Web Tokens for Enhancing User Security," vol. 5, no. 4, pp. 5309–5312, 2024.

[43] S. Naik, R. D. Dandagwhal, C. N. Wani, and S. K. Giri, "A review on various aspects of auxetic materials," *AIP Conf. Proc.*, vol. 2105, no. 05, pp. 90–92, 2019, doi: 10.1063/1.5100689.