Kandyba Ihor, PhD, Senior Lecturer of the Department of Software Engineering, Petro Mohyla Black Sea National University, ORCID: https://orcid.org/0000-0002-8589-4028

Horban Hlib, Candidate of Technical Sciences (PhD), Associate Professor, Associate Professor of the Department of Software Engineering, Petro Mohyla Black Sea National University, ORCID: https://orcid.org/0000-0002-6512-3576

Fisun Mykola, Doctor of Technical Sciences, Professor, Professor of the Department of Software Engineering, Petro Mohyla Black Sea National University, ORCID: ORCID: https://orcid.org/0000-0003-1297-6230

PYTHON-BASED TOOLS FOR WEB-ORIENTED MANAGEMENT OF A WINDOWS-BASED COMPUTER

Abstract

This paper investigates the capabilities of the Python programming language for implementing web-based control of the Windows operating system and proposes an implementation of an information system that enables such control. An analysis of recent publications in the field of Windows system administration is conducted, and the main research directions in this area are identified. The study determines the set of tools required for implementing a web interface in applications developed using Python. The features of the Django web framework are described in the context of remote computer control. Python libraries for analyzing the Windows operating system and computer hardware are examined, and their advantages and disadvantages are identified and compared. The role of the Windows Event Log in the process of Windows system administration is outlined, and Python tools for interacting with Windows Event Log are analyzed. A web interface for visualizing these logs is presented. The integration of Large Language Models, specifically the Gemini 3 Flash model, for

analyzing Windows Event Log data is proposed. The process of integrating this model into the developed remote control system is described. The Windows Registry and its role in software management within the Windows operating system are also examined. Python tools for working with the Windows Registry are reviewed, and the process of uninstalling installed software using registry keys is described. Furthermore, the specifics of launching additional software using the subprocess module are investigated. The capabilities of the Winget package manager are analyzed, along with its features in the Windows environment. An implementation of package list updating and software installation using a combination of Python tools and Winget is presented. Finally, potential directions for further development of the proposed information system for remote control of computers running the Windows operating system are identified, including the integration of advanced data analysis tools and artificial intelligence technologies: TensorFlow, PyTorch and Waikato Environment for Knowledge Analysis.

Keywords: Python, Django, Windows Event Log, Windows Registry, Software development.

Problem statement. Management and administration of a computer network running the Windows operating system requires specialized tools. The most widespread solution is Active Directory, which enables remote configuration of the operating system, user management, software deployment, and more. However, tasks such as monitoring the use of computational resources and processing event logs require additional tooling.

Most often, remote access protocols such as Secure Shell (SSH), Remote Desktop Protocol (RDP), and Virtual Network Computing (VNC) are used to address these tasks. Nevertheless, the use of remote access protocols for remote management has certain drawbacks, including the lack of capabilities for viewing

the history of resource monitoring, the need for additional software for data analysis, and the requirement to install extra cross-platform tools.

In operating systems based on the Linux kernel, web-oriented administration tools (e.g., Webmin) are widely used [1, p. 80], providing centralized management, monitoring, and event history storage through a web browser. Similar applications also exist for Windows; however, they are less widespread and are often focused on managing server applications and services. Meanwhile, the primary administration tools for Windows remain RDP, PowerShell, and local graphical utilities. The use of the Python programming language makes it possible to monitor the utilization of a computer's computational resources, implement management functions, analyze activity logs, and develop a web interface to provide access to this functionality.

Analysis of recent research and publications. In the field of Windows operating system administration, the study of the applicability of Large Language Models (LLMs) for activity log analysis is of particular relevance. In study *Karlsen, E., Luo, X., Zincir-Heywood, N., Heywood, M.* [2, p. 35], the authors propose a specialized web application that leverages LLMs to detect system malfunctions and generate recommendations for their resolution and for improving OS performance. The authors also analyze the effectiveness of LLM-generated recommendations and their interpretability by users. Although the study demonstrates the potential of LLMs to simplify the administration of Windows-based computers, the issue of implementing this approach through a web interface requires further investigation.

In paper *Sudharson, K., Balaji, S., Deepak Reddy, A., Sai Ram, V.* [3, p. 435], the Remote Desktop Protocol (RDP), which is used by Windows by default, is examined. The authors present the main advantages and disadvantages of this protocol in an educational context. In addition, an alternative protocol for connecting to a remote computer is proposed, which allows multiple users to

connect to a single desktop and results in lower network traffic consumption. However, the use of the proposed solutions via a web interface remains an open question requiring further research.

A relevant research direction in system administration is the use of Security Information and Event Management (SIEM) systems. Publication *Vielberth, M.* [4, p. 2304] describes the application of these systems for analyzing Windows OS activity logs in order to detect malware, identify software and hardware malfunctions, and diagnose network issues. The article demonstrates the feasibility of analyzing collected data from Windows activity logs; however, issues related to monitoring the use of computational resources and remote computer management require further study.

The conducted analysis indicates that remote management tools and activity log analysis are important in the field of computer administration. However, the integration of remote management tools with data analysis of computational resource usage and Windows OS activity logs remains an open problem requiring further investigation.

Formulation of the article's objectives. The aim of this study is to develop an information system for web-based computer management supporting the Windows operating system using Python tools. The development of such a system will simplify the administration of computer networks running Windows OS.

The main tasks to be addressed in order to achieve this goal are:

1) identification of Python tools for analyzing hardware and the Windows operating system;

2) analysis of tools for developing web interfaces for applications implemented in the Python programming language;

3) investigation of the possibilities of installing and uninstalling software using Python;

4) identification of prospective directions for the development of a web-based management system.

Main part. Tasks related to monitoring the usage of computational resources and analyzing event logs require the use of specialized software tools, as remote access protocols SSH, RDP, and VNC provide only the capability to connect to a remote system, but are not designed for collecting, storing, and analyzing historical data on resource states. Therefore, effective monitoring and event processing typically rely on dedicated tools, which may require the installation of additional software.

The development of an information system for web-based computer management is impossible without tools for monitoring the use of computational resources. The Python programming language provides several libraries that implement this functionality in the Windows operating system, including psutil, system_info, cpuinfo, wmi, and others. Each library is designed to analyze a specific component:

– *system_info* allows determining the characteristics of the operating system. When developing a computer management application, it is crucial to accurately identify the OS kernel in order to use appropriate implementation tools. For example, wmi is designed exclusively for Windows and cannot be used with other operating systems.

– *cpuinfo* is a library that provides detailed information about the computer's central processing unit. It outputs data such as the model, clock frequency, supported protocols, architecture, and cache size. The library is cross-platform, enabling the application to be used on Linux-based operating systems, which in turn facilitates further development of web-based management systems.

– *psutil* is a cross-platform library for monitoring OS resource usage, including current CPU load, total and available RAM, disk storage capacity (used

and free), network speed, and more. In addition, it allows process management and access to data from built-in sensors.

– *WMI* is a library that enables interaction between Python and the Windows Management Instrumentation framework. This framework collects information about all connected devices and allows process management and modification of system settings. WMI is based on an object-oriented model, where all information about the computer is represented as classes.

The listed tools make it possible to implement monitoring of the Windows OS. However, presenting the obtained information requires the use of an interface. Python supports several specialized frameworks for implementing such interfaces, including Flask, FastAPI, and Django. In the context of developing an interface for a web-based computer management information system, Django is the most appropriate choice.

Django provides Object-Relational Mapping (ORM) for database interaction, an authentication system, user session management mechanisms, protection against common vulnerabilities (such as SQL injection and CSRF attacks), and an automatically generated administrative panel.

Web pages are rendered using templates and a dedicated markup language known as the Django Template Language (DTL) [5, p. 34]. This is a text-based language with syntax similar to standard HTML but enhanced with special constructs that allow adding logic, such as displaying variables, creating loops, and using conditional statements.

When implementing a web-based monitoring system, dynamic page generation is required to display the current level of computational resource usage, hardware information, operating system activity logs, and the results of software installation and removal operations (Fig. 1).

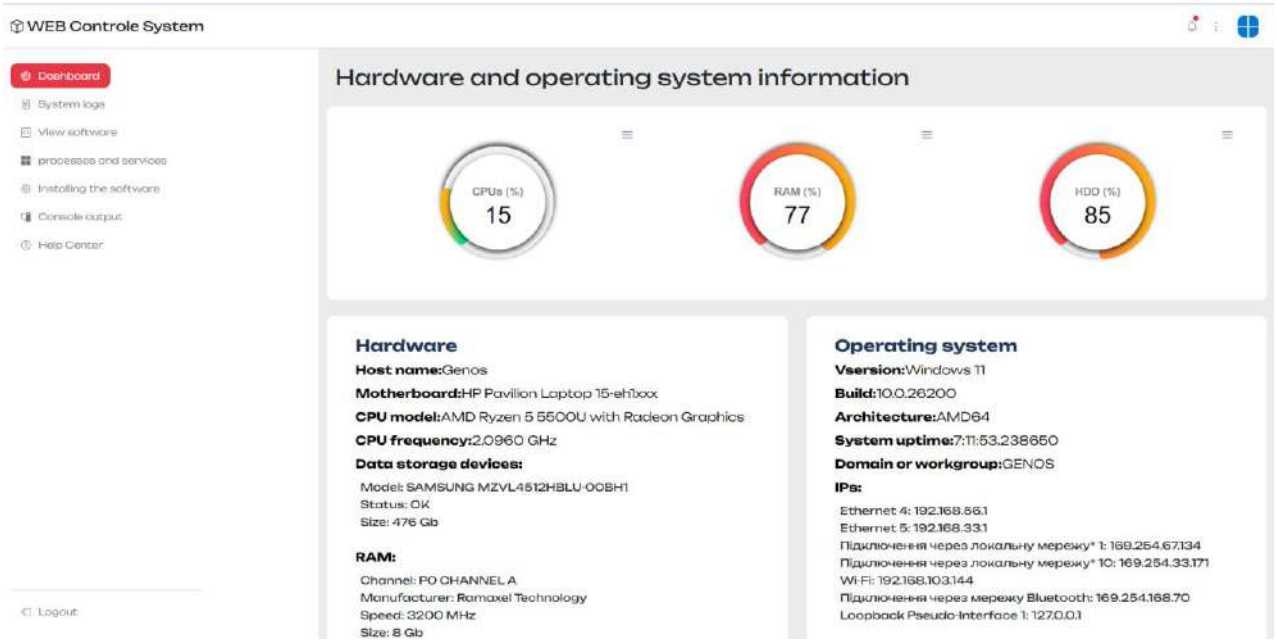


Fig. 1. Interface of the Information System for Web-Based Computer Management Running the Windows Operating System

In addition to displaying the current usage of computational resources, a web-based management information system should provide the ability to view operating system activity logs. In Windows OS, these logs are referred to as the Windows Event Log (WEL) and can be read and analyzed using the Python programming language. WEL is a centralized event logging system that contains records of Windows OS operations, security-related events (such as login attempts and file access), and the results of other software execution. For administrators, these logs are a critical tool, as they enable in-depth security auditing, timely diagnosis of system issues, and tracking of user activity within the network.

The Pywin32 library enables interaction between Python and the built-in Windows tools for working with WEL. Using Pywin32 is significantly faster [6] than processing exported WEL data and does not require executing additional commands via subprocess. This makes it possible to display current logs through a web interface (Fig. 2) and integrate additional processing tools, such as Large Language Models LLMs).

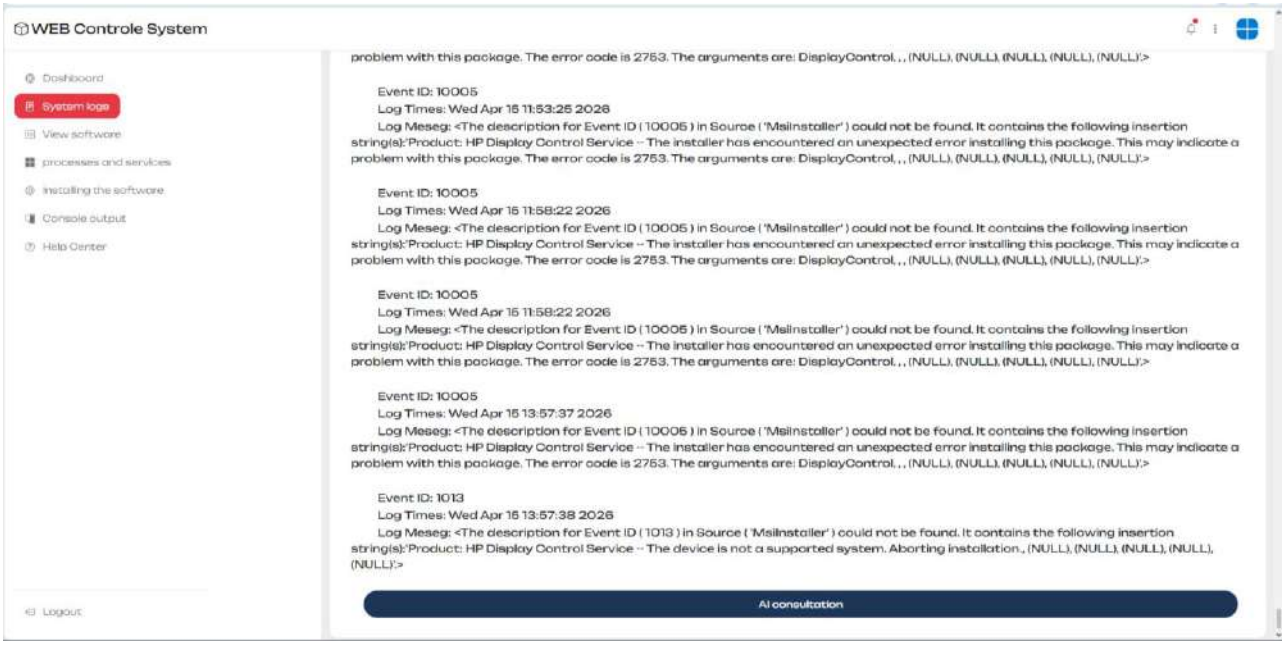


Fig. 2. Interface for Viewing the WEL

A large number of records in the Windows Event Log (WEL) complicates the process of diagnosing the Windows operating system. To simplify this process, the capability of obtaining assistance from AI tools, such as Large Language Models (LLMs), has been implemented. LLMs are a type of AI technology based on neural network architectures, trained on large volumes of textual data to understand and analyze user queries. The Python programming language enables integration with LLMs of various architectures, including Gemini 3 Flash, which is one of the largest and fastest models available on the modern market [7].

Interaction with Gemini 3 Flash in Python is implemented by connecting the google-generativeai library and generating requests using the generate_content() method, with prior model selection via GenerativeModel(). However, to use this model, it is necessary to create and configure a dedicated project in the Google AI Studio service. This service not only allows configuration of API keys for integrating Gemini 3 Flash or other Gemini models but also supports LLM testing and comparative analysis.

Management of the Windows operating system necessarily includes the ability to install and uninstall software. An information system built with Python can use several approaches to obtain information about installed software: WMI tools, data from the Windows Registry, or launching separate applications via subprocess. The disadvantage of WMI lies in its excessive consumption of computational resources, as queries to built-in classes such as Win32_Product trigger a full integrity check of each MSI package. At the same time, data obtained via subprocess requires additional processing and may lead to encoding conflicts (for example, between systems using UTF-8 and CP-1251). Moreover, both approaches create a significant load on the operating system due to the execution of additional processes or services, making them less stable and secure compared to direct access to the Windows Registry [8, pp. 50].

Accessing Windows Registry data in Python can be achieved using the winreg library. This is a built-in Python module that provides direct access to the Windows Registry API and supports the creation and modification of keys and their values. Due to this direct access, winreg enables fast retrieval of installed software lists, management of startup programs, and modification of user system settings.

The Windows Registry section *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall* contains information about applications installed in the Windows OS. In this section, a subkey is created for each application, typically named as a GUID or the application name. Each subkey includes a set of parameters such as DisplayName (application name), DisplayVersion (version), Publisher (vendor), InstallDate (installation date), and UninstallString (command for uninstallation). These data are used by the system and third-party tools to display the list of installed software in the “Programs and Features” control panel and can also be utilized for software removal through a web-based computer management system (Fig. 3).

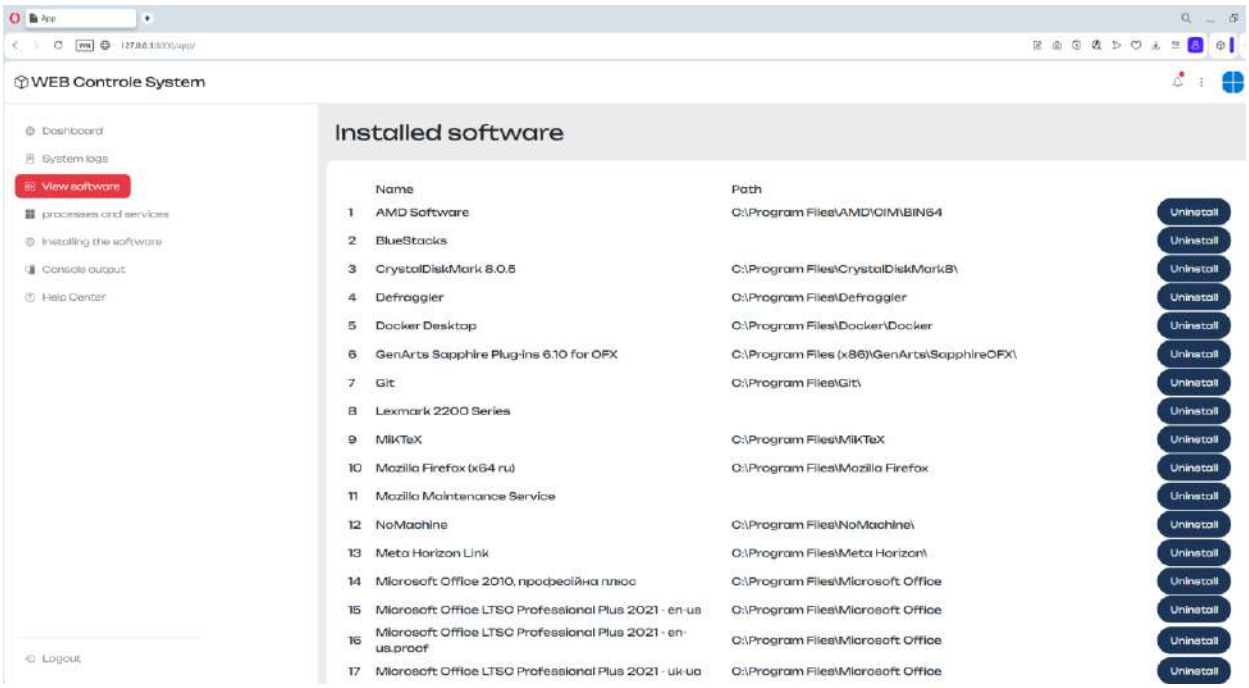


Fig. 3. Interface for viewing installed software

The QuietUninstallString subkey in the Windows Registry contains a command for “silent” software uninstallation without user interaction. This method, unlike UninstallString, which typically launches a standard uninstaller with a graphical interface, involves the use of specific parameters (e.g., /quiet, /silent, /qn) that allow the process to run in the background. This approach to uninstallation is essential for implementing web-based management.

The subprocess module of the Python programming language provides the ability to execute OS commands, which makes it possible to use the value of QuietUninstallString to uninstall software via a web-based management system. The QuietUninstallString is passed to functions such as subprocess.run(), ensuring the execution of the command specified in this parameter.

Software installation can be implemented using the winget tool. This tool, also known as Windows Package Manager (winget) [9, pp. 35], is an official package management solution in the Microsoft Windows operating system, developed by Microsoft to simplify the installation, updating, and removal of

software via the command line. It enables automation of software management by using a centralized application repository and standardized commands such as install, upgrade, list, and uninstall.

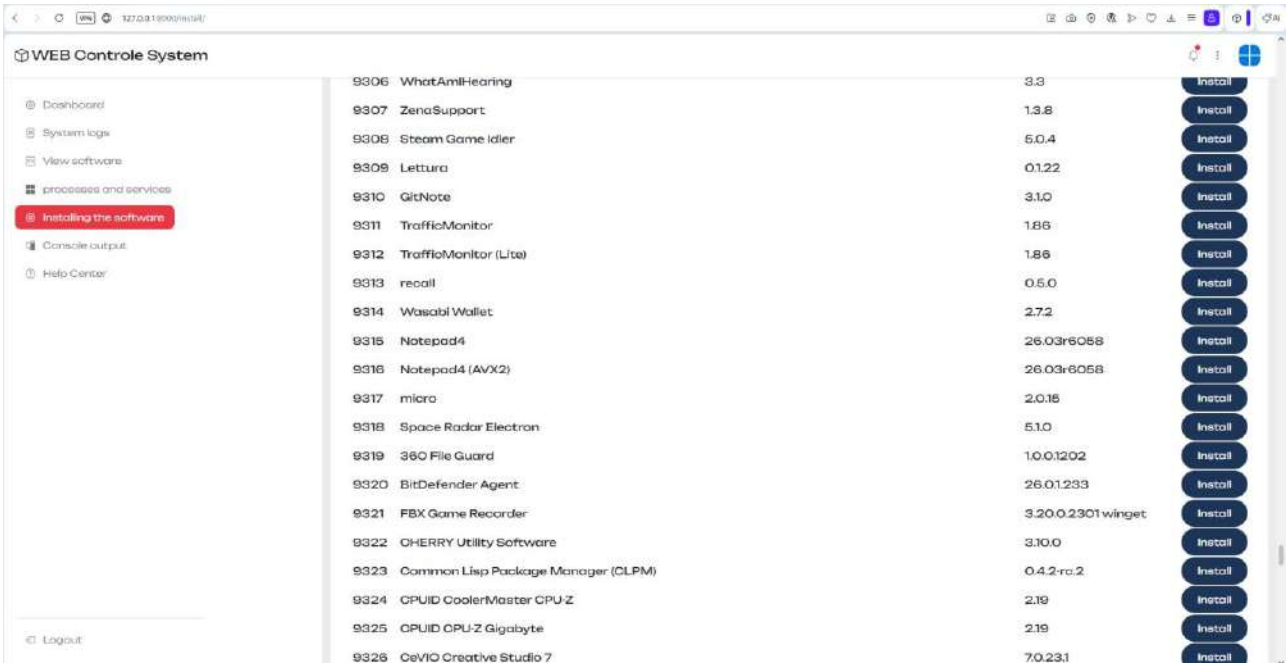


Fig. 4. Interface for viewing available software in Winget

An implicit advantage of using Winget is the reduced risk of installing malicious software. This tool installs applications from a centralized repository where software is verified and validated. When using Winget, installation does not require downloading third-party installers, which, when obtained from external sources, often conceal malicious software. In addition, Winget simplifies the software management process, helping to avoid the use of outdated versions with known vulnerabilities.

However, the use of Winget does not guarantee complete security. There is always a risk of installing unwanted or potentially harmful software, as well as possible supply-chain threats. Therefore, even when using Winget, it is still necessary to employ specialized security software to protect the computer system.

In future work, to improve the process of diagnosing software and hardware issues, it is planned to integrate additional Python tools such as TensorFlow and PyTorch, which will enable the development of more advanced machine learning models. Furthermore, by leveraging the built-in ORM and connecting an additional database, it becomes possible to implement analysis of the history of computational resource usage. Such analysis can be performed using various approaches, including the previously mentioned LLMs, TensorFlow, PyTorch, or by utilizing the WEKA toolkit.

The WEKA platform (Waikato Environment for Knowledge Analysis) is a representative open-source software suite designed to automate data mining and machine learning processes [10 p. 21]. Its relevance lies in the consolidation of an extensive library of modern machine learning algorithms, statistical modeling techniques, and methods for evaluating the adequacy of obtained results. The use of WEKA will make it possible to analyze the history of collected measurements of computational resource usage, WEL records, and to identify potential issues based on this data.

Conclusions

1) An analysis of Python tools that enable the inspection of hardware and the characteristics of the installed Windows operating system was conducted. Three of the most suitable tools were selected: system_info for retrieving operating system information, cpuinfo for obtaining central processing unit specifications, Psutil for monitoring implementation, and WMI for determining the characteristics of the GPU, RAM, and storage devices.

2) Tools for implementing a web interface in applications developed using the Python programming language were analyzed. The web framework with the highest level of functionality–Django was selected. Its capabilities are briefly described in the context of developing tools for web-based computer control.

3) The possibilities of installing and uninstalling software in the Windows operating system using Python and Winget were investigated. The capability of viewing WEL via a web interface and their subsequent analysis was also examined.

4) Directions for improving the developed web-oriented information system for computer control were proposed. The potential integration of more advanced artificial intelligence tools–TensorFlow, PyTorch, and the WEKA platform–was considered.

References

1) Westfall, J. (2021). Basic Linux Administration via GUI (Webmin). In Set Up and Manage Your Virtual Private Server: Making System Administration Accessible to Professionals (pp. 77-110). Springer.

2) Karlsen, E., Luo, X., Zincir-Heywood, N., & Heywood, M. (2024). Benchmarking Large Language Models for Log Analysis, Security, and Interpretation. Journal of Network and Systems Management, 32(3), 59. https://doi.org/10.1007/s10922-024-09831-x

3) Sudharson, K., Balaji, S., Deepak Reddy, A., & Sai Ram, V. (2022). Speedy and secure remote management protocol using virtualization. International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 3, 435-442. https://doi.org/10.1007/978-981-19-3679-1_35

4) Vielberth, M. (2025). Security Information and Event Management (SIEM). In S. Jajodia, P. Samarati, & M. Yung (Eds.), Encyclopedia of Cryptography, Security and Privacy (pp. 2304–2306). Springer Nature Switzerland. https://doi.org/10.1007/978-3-030-71522-9_1681

5) Mele, A. (2020). Django 3 By Example: Build powerful and reliable Python web applications from scratch. Packt Publishing Ltd.

6) Studiawan, H., Sohel, F., & Payne, C. (2020). Automatic Event Log Abstraction to Support Forensic Investigation. ACSW '20: Australasian Computer Science Week 2020. ACM. DOI:https://doi.org/10.1145/3373017.3373018

7) Zhang, P., Wang, J., Hu, X., Wang, X., Fan, X., Chi, W., & Yang, W. (2026). Comparative performance of GPT-4, GPT-o3, GPT-5, Gemini-3-Flash, and DeepSeek-R1 in ophthalmology question answering. Frontiers in Cell and Developmental Biology, Volume 14. https://doi.org/10.3389/fcell.2026.1744389

8) Halsey, M., & Bettany, A. (2015). Windows registry troubleshooting. Apress.

9) Ifrah, S. (2024). Install Tools (Azure CLI, Azure PowerShell, VS Code, Copilot). In Getting Started with Azure OpenAI: Deploying and Managing Azure AI and Azure OpenAI Solutions (pp. 33–79). Springer.

10) Aher, S. B., & Lobo, L. (2011). Data mining in educational system using Weka. In Proceedings of the International Conference on Emerging Technology Trends (ICETT) (pp. 20-25).