Journal on Informatics Visualization and Social Computing
Volume 1, Issue 1, 01-08
e_ISSN: 3123-7002
https://e-journal.gomit.id/jivsc
DOI: 10.58723/jivsc.v1i1.44

Design and Development of An Android-Based Server Monitoring Application Using The aaPanel API with The Waterfall Method

Teguh Rijanandi
Telkom University,
INDONESIA

Eko Risdianto
University of Bengkulu,
INDONESIA

Mohammad Qais Rezvani
Kurukshetra University
INDIA

*** Corresponding author:**
Teguh Rijanandi, Telkom University, INDONESIA. ✉ teguhnandi@student.telkomuniversity.ac.id

Article Info

Article history:

Received: July 14, 2025
Revised: July 22, 2025
Accepted: July 23, 2025

Keywords:

aaPanel
Mobile Application
React Native
Server Monitoring
Waterfall Method

Abstract

ABSTRACT
Background of study: The evolution of cloud computing has transformed server management, yet many management interfaces remain web-based and are not optimized for mobile devices. aaPanel, a popular server control panel, also faces this challenge of providing a mobile-friendly monitoring experience.
Aims and scope of paper: This research aims to design and build a mobile application named "aaPanel Mobile" to efficiently monitor vital server statistics on Android devices, addressing the gap in mobile accessibility for aaPanel users.
Methods: The software development follows the Waterfall model, covering requirements, design, development, testing, deployment, and maintenance. The application was built using Expo React Native, integrates with the aaPanel API for data retrieval, and its functionality was verified through black-box testing.
Result: Testing results confirmed that all primary functionalities were successfully implemented. The application correctly displayed resource usage statistics, provided real-time data updates, and listed websites as expected.
Conclusion: The "aaPanel Mobile" application has met its design and functional objectives, proving to be a viable tool ready for use in a production environment to help administrators monitor servers from anywhere.

To cite this article: Rijanandi, Risdianto, Rezvani (2025). Design and Development of An Android-Based Server Monitoring Application Using The aaPanel API with The Waterfall Method. *Journal on Informatics Visualization and Social Computing, 1*(1), 01-08. https://doi.org/10.58723/jivsc.v1i1.44

INTRODUCTION

Information technology plays a crucial role in various human activities and has become a main driver in the business world. As a result, information technology fundamentally transforms how organizations operate and are structured, to support vital functions such as the collection, processing, generation, storage, retrieval, and transmission of information (Faittullah et al., 2023). In line with this, the advancement of information and communication technology has also driven digital transformation across various sectors, where cloud computing stands as one of its main pillars. Cloud computing allows users to access and manage computing resources such as servers, storage, and databases over the internet with high flexibility and scalability (T. Hidayat & Mahardiko, 2020). To simplify the management of these resources, various server control panel software have been developed. One of the popular control panels among developers and system administrators is aaPanel. aaPanel is an **open-source** control panel that provides a simple yet powerful graphical user interface (GUI) for managing servers, websites, databases, and other services (Jamaluddin et al., 2024). The panel offers one-click features such as LNMP/LAMP environment setup and software installation, aiming to streamline server deployment. Its primary goal is to save users time on system configuration, enabling them to focus more on their core development projects (Arman, 2025). As shown in Figure 1, the main aaPanel dashboard presents a comprehensive system status summary,

displaying vital metrics like CPU usage, RAM usage, and server load status in intuitive circular graphs. Additionally, the dashboard provides an overview of the number of active sites, databases, and security configurations.

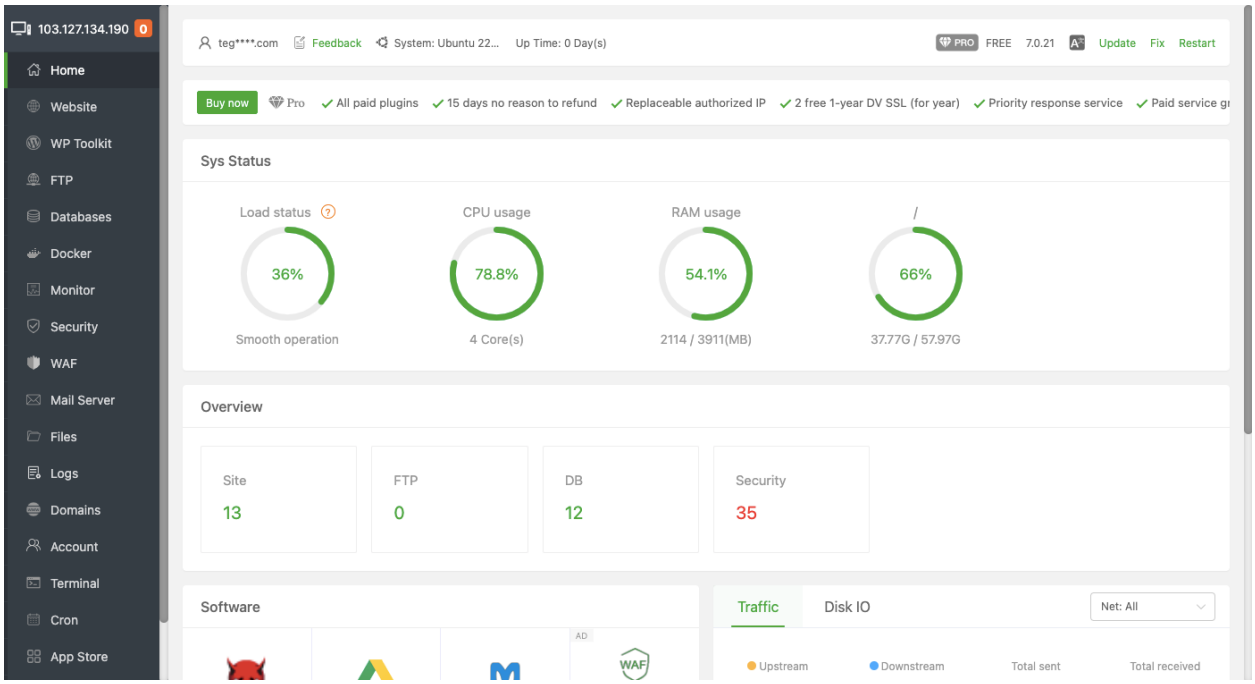

Figure 1. aaPanel Dashboard Interface

Although highly functional, the web-based interface of aaPanel is primarily designed for desktop access. When accessed from a mobile device, users often need to pinch-and-zoom, which reduces comfort and efficiency in monitoring. To address this limitation, this research focuses on the design and development of "aaPanel Mobile," an **Android-based** application that acts as a client for monitoring servers managed by aaPanel.

METHOD

The software development methodology used in this research is the Waterfall model. This model was chosen for its structured and systematic workflow (Bassil, 2022; Yulianti et al., 2022; Rijanandi et al., 2023). The Waterfall methodology was selected due to its structured and linear nature, which aligns well with the stable and clearly defined requirements of the server monitoring application. Its emphasis on comprehensive documentation and well-defined deliverables ensured that each stage of development from requirements analysis to deployment could be executed with clarity, consistency, and predictability. This approach proved effective for integrating the aaPanel API and delivering a mobile solution ready for practical use (Pargaonkar, 2023). Waterfall is well-suited for projects where the requirements are clearly defined and unlikely to change (Diansyah et al., 2023).

The following are the stages of the Waterfall model applied in this research:

1. Requirements: The initial phase to identify and document all functional and non-functional requirements of the application to be built (Rijanandi et al., 2022a). The requirements stage plays a very important role, as it serves as the main foundation for the entire subsequent development process (Sulfikar et al., 2024).
2. Design: The phase for designing the system architecture, user interface (UI), and user experience (UX) based on the defined requirements (Fandi et al., 2023).
3. Development: During the development phase, the project is divided into smaller components, which are then built and integrated step by step until the complete product is finalized (Mishra & Ibrahim, 2023).
4. Testing: The phase of testing the application to find and fix bugs or errors, and to ensure the application meets all specified requirements (Cahyono et al., 2022).

5. Deployment: The phase of releasing the application to end-users, in this case, packaging it into an APK format for the Android platform (Rijanandi et al., 2022b).
6. Maintenance: The post-release maintenance phase, which includes bug fixes, updates, and the addition of future features.

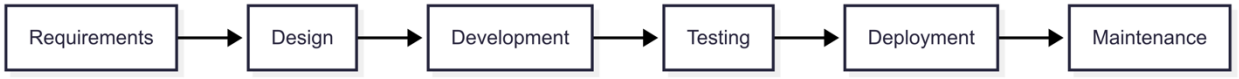

Figure 2. Waterfall Method Flow (Sanmocte & Costales, 2025).

In the testing phase, this research uses the black-box testing method. This testing focuses on the application's functionality without regard to its internal code structure. The testing scenarios used to verify the application's functionality are summarized in the following table:

Table 1. Black-Box Testing Scenarios

No.	Testing Scenario
1.	Can the application display server resource usage statistics (CPU, Memory, Disk)?
2.	Can the application display statistics in real-time or near real-time?
3.	Can the application display the list of websites on the server?

RESULTS AND DISCUSSION

This section describes the results from each stage of the implemented Waterfall method.

3.1. Requirements

Based on the research objectives, the main functional requirements for the "aaPanel Mobile" application were identified as follows:
Users must be able to log in using their aaPanel server credentials.

1. The application must display real-time CPU usage.
2. The application must display real-time Memory (RAM) usage.
3. The application must display storage (Disk) usage.
4. The application must display basic server system information.
5. The application must display a list of websites hosted on the server.

3.2. Design

In this phase, the user interface (UI) and user experience (UX) were designed. The design focused on presenting complex statistical data in a clean, simple, and easy-to-read format on a mobile screen. As shown in Figure 3, the application interface design consists of three main screens accessible via a bottom navigation bar.

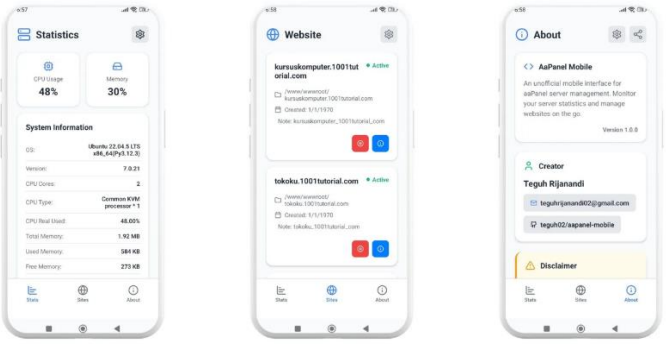

Figure 3. Application Interface Design

A description of each screen is as follows:

1. Statistics Screen: This is the main screen that greets the user. The top section displays a visual summary of CPU and Memory usage in an easy-to-read percentage format. Below it, the "System Information" section presents detailed technical server information, such as OS version, number of CPU cores, total memory, and used and free memory.
2. Website Screen: This screen is for managing websites on the server. Each website is displayed as a card containing important information such as the domain name, status (e.g., "Active"), and creation date. There are also quick action buttons for basic management.
3. About Screen: This screen contains information about the application itself, including a brief description, version number, creator details, and links to contacts and the code repository. There is also a "Disclaimer" section for legal or affiliation information.

Overall, the design uses a minimalist approach with clear iconography and a card-based layout to ensure information can be accessed quickly and efficiently.

3.3. Development

The application was developed according to the specifications from the design phase. The technology used is Expo React Native, a framework that allows for cross-platform application development using JavaScript and React. React Native, an open-source framework created by Meta, streamlines the process of building mobile applications by offering a unified development environment for both iOS and Android platforms without the need for WebView integration (Liu et al., 2025). React Native leverages native UI components the very elements used in traditional native applications enabling it to deliver a faster and more seamless user experience (Goli, 2021).

The interface design utilizes an API (Application Programming Interface), which is a set of protocols created by system developers to allow certain parts or the entirety of a system's functions to be accessed programmatically (A. T. Hidayat et al., 2021). In terms of API communication, the application adopted asynchronous requests to ensure non-blocking interactions with the server. This technique improves performance and enhances user experience by preventing application freeze during data fetch operations. Additionally, error-handling mechanisms were implemented to manage issues such as invalid API keys, network timeouts, or server errors, ensuring the system remains stable during runtime. To retrieve data from the server, the application integrates the aaPanel API. Interaction with the API begins with a secure authentication process. As seen in Figure 4, an AaPanelApi class was created to manage the connection. This class requires a **panel_url** and **api_key** upon initialization. This mechanism ensures that every request to the server must include a valid API key, thereby securing access to server data.

```
/**
 * Initializes the API client.
 * @param {string} panel_url - The base URL of your aaPanel instance (e.g., "https://192.168.1.245:7800").
 * @param {string} api_key - The API key from your panel's settings page.
 */
constructor(panel_url, api_key) {
    if (!panel_url || !api_key) {
        throw new Error("Panel URL and API Key are required.");
    }
    this.PANEL_URL = panel_url;
    this.API_KEY = api_key;

    // In-memory storage for the session cookie.
    // For persistence in React Native, you could extend this to use AsyncStorage.
    this.cookie = null;
}
```

Figure 4. API Initialization Code Snippet

Once the connection is successfully established, the application can retrieve specific data by calling predefined functions. Figure 5 shows several functions such as **getSystemTotal()**, **getDiskInfo()**,

and **getNetWork()**. Each of these functions is responsible for making a request to the corresponding API endpoint on the aaPanel server to fetch system statistics, disk information, and network traffic. This approach makes the code more structured and manageable.

```
/**
 * Gets basic system statistics (OS, CPU, Memory).
 * @returns {Promise<object|null>} System statistics object.
 */
async getSystemTotal() {
    return this._request('/system', { action: 'GetSystemTotal' });
}

/**
 * Gets disk partition information.
 * @returns {Promise<object|null>} Disk information array.
 */
async getDiskInfo() {
    return this._request('/system', { action: 'GetDiskInfo' });
}

/**
 * Gets real-time status (CPU, memory, network, load).
 * @returns {Promise<object|null>} Network and load statistics.
 */
async getNetWork() {
    return this._request('/system', { action: 'GetNetWork' });
}
```

Figure 5. Statistical Data Fetching Code Snippet

The application calls these functions periodically to update the data displayed on the interface, thus providing the user with the illusion of real-time monitoring.

3.4. Testing

In the system testing phase, the Black-box testing method is applied to detect potential issues and ensure that all components of the system operate as intended (Rumetna et al., 2022). Black-box testing was conducted based on the scenarios established in the research methodology section. In addition to internal testing, user acceptance testing was also carried out involving several potential users. As shown in Figure 6, these testing sessions were conducted remotely via Zoom, where users were asked to perform tasks according to the test scenarios and provide direct feedback on the application's functionality and usability.

Figure 6. User Testing Session via Zoom

The results from all tests are presented in the following table:

Table 2. Functionality Testing Results

No.	Testing Scenario	Result	Remarks
1.	Displaying resource usage statistics (CPU, Memory, Disk).	Success	The application was able to display data in graphical and text formats.
2.	Displaying statistics in real-time.	Success	The application automatically refreshes data at set intervals.
3.	Displaying the list of websites on the server.	Success	The application was able to fetch and display the list of websites.

The test results show that all core functionalities tested were successful and met the established requirements.

3.5. Deployment

After the application successfully passed the testing phase, the next step was deployment. This process aims to package the application into a file that can be distributed and installed by users on Android devices. For this purpose, the Expo Application Services (EAS) Build service was used. As shown in Figure 7, the Expo dashboard displays the history of successful build processes, resulting in an APK (Android Package Kit) file ready for distribution.

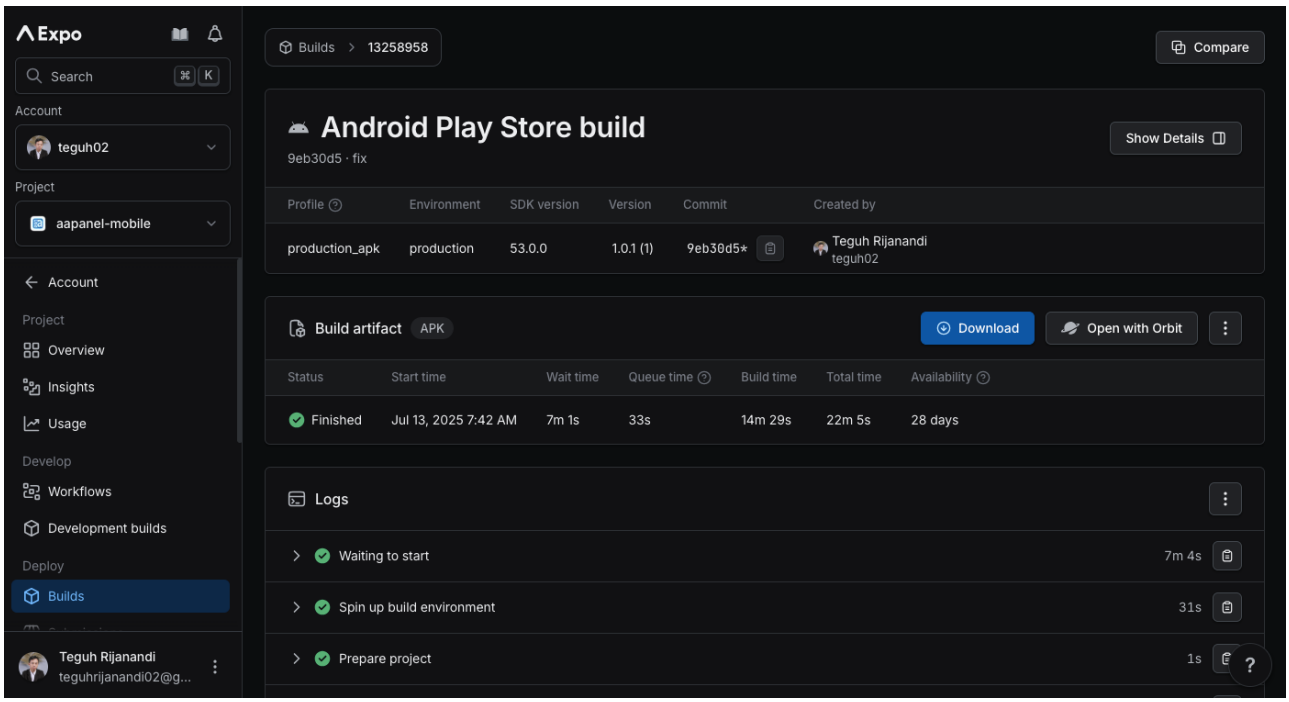

Figure 7. Application Build Process Using Expo

The successfully built application was then uploaded to a public repository for easy access. Users can download the latest version of the application from the releases page at the following link: https://github.com/teguh02/aapanel-mobile.

3.6. Maintenance

The final phase in the Waterfall method is maintenance. After the application's release, a maintenance plan was established to ensure its functionality and security are preserved. The primary plan is to perform regular updates. These updates will focus on adapting the application to any

changes or developments in the aaPanel API. Additionally, this phase includes fixing bugs that may be discovered by users in a production environment, as well as potentially adding new features based on feedback from the user community.

CONCLUSION

From the test results, it can be concluded that the application has met all the criteria set in the testing phase. The application can display server resource usage statistics, display statistics in real-time, and display the list of websites on the server. Thus, the "aaPanel Mobile" application is ready to be implemented and used in a production environment as an effective tool for server administrators to perform monitoring anytime and anywhere.

ACKNOWLEDGMENT

The author would like to express sincere gratitude to all parties who have provided support and contributions to this research, whose names cannot be mentioned one by one. Their assistance was invaluable to the completion of this study.

AUTHOR CONTRIBUTION STATEMENT

Teguh Rijanandi: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing – Original Draft, Writing – Review & Editing, Visualization, Project Administration. Eko Risdianto: Validation, Investigation, Writing – Review & Editing.

REFERENCES

Arman, M. (2025). Rancang Bangun Web Server Blog Dengan Layanan Vps Dan Navigasi. *Jurnal Ilmiah BETRIK (Basemah Teknologi Informasi & Komputer)*, *16*(01), 59–73. https://doi.org/10.36050/w5xy6d50

Bassil, Y. (2022). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Software Engineering (IJSE*, *2*(5), 1–11. https://doi.org/255895133318216

Cahyono, T., Setianingsih, S., & Iskandar, D. (2022). WEBSITE-BASED BOOK LENDING SYSTEM IMPLEMENTASI METODE WATERFALL PADA PERANCANGAN SISTEM. *Jurnal Teknik Informatika (JUTIF)*, *3*(3), 726. https://doi.org/10.20884/1.jutif.2022.3.3.285

Diansyah, A. F., Rahman, M. R., Handayani, R., Cahyo, D. D. N., Utami, E., Info, A., & Development, S. (2023). Comparative Analysis of Software Development Lifecycle Methods in Software Development: A Systematic Literature Review. *International Journal of Advances in Data and Information Systems*, *4*(2), 104. https://doi.org/10.25008/ijadis.v4i2.1295

Fandi, F., Imaniawan, D., Pratmanto, D., Rijanandi, T., & Silvia, A. (2023). Designing An Animal Adoption and Social Media Information System Using The SDLC Waterfall Method. *Jurnal Tekno Info*, *17*(ue 1)). https://doi.org/10.33365/jti.v17i1.2333

Goli, V. R. (2021). React Native Evolution, Native Modules, and Best Practices. *International Journal of Computer Engineering and Technology (IJCET)*, *12*(2), 73–85. https://doi.org/10.34218/IJCET_12_02_009

Hidayat, A. T., Saputra, W. A., & Saifullah, S. (2021). Website-Based E-Pharmacy Application Development to Improve Sales Services Using Waterfall Method. *International Journal of Advances in Data and Information Systems*, *2*(2), 116. https://doi.org/10.25008/ijadis.v2i2.1226

Hidayat, T., & Mahardiko, R. (2020). A Systematic Literature Review Method On AES Algorithm for Data Sharing Encryption On Cloud Computing. *International Journal of Artificial Intelligence Research*, *4*(1). https://doi.org/10.29099/ijair.v4i1.154

Jamaluddin, Muhammad Zamroni Uska2, Rasyid Hardi Wirasasmita3, M. R. (2024). JITE (Journal of Informatics and Telecommunication Engineering) Development of Smart Servers for Informatics Education Program Using NDLC Method. *JITE (Journal of Informatics and Telecommunication Engineering)*, *7*(January), 597–606.

https://doi.org/10.31289/jite.v7i2.10853
Liu, Y., Chen, X., Liu, P. E. I., Samhi, J., Grundy, J., Chen, C., & Li, L. I. (2025). Demystifying React Native Android Apps for Static Analysis. *ACM Transactions on Software Engineering and Methodology*, *34*(4), 3. https://doi.org/10.1145/3702977
Mishra, A., & Ibrahim, Y. (2023). Structured software development versus agile software development : a comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504–1522. https://doi.org/10.1007/s13198-023-01958-5
Muhammad Faittullah Akbar, A. F. (2023). Application of Waterfall Method In Design Of Web-Based Library Information System Program Case Study at Elementary School Warungnangka Kabupaten Subang. *Jurnal Teknologi Dan Open Source*, *6*(1), 72. https://doi.org/10.36378/jtos.v6i1.3065
Pargaonkar, S. (2023). A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and. *International Journal of Scientific and Research Publications*, *13*(8), 120–124. https://doi.org/10.29322/IJSRP.13.08.2023.p14015
Rijanandi, T., Dimas, T., Wibowo, C. S., Pratama, I. Y., Dharma Adhinata, F., Utami, A., & Studi, P. (2022a). Web-Based Application with SDLC Waterfall Method on Population Administration and Registration Information System (Case Study: Karangklesem Village, Purwokerto. *Jurnal Teknik Informatika (JUTIF)*, *3*(1), 99–104. https://doi.org/10.20884/1.jutif.2022.3.1.145
Rijanandi, T., Dimas, T., Wibowo, C. S., Pratama, I. Y., Dharma Adhinata, F., Utami, A., & Studi, P. (2022b). Web - Based Application with SDLC Waterfall Method on Population Administration and Registration Information System (Case Study: Karangklesem Village, Purwokerto. *Jurnal Teknik Informatika (JUTIF*, *3*(1), 99–104. https://doi.org/10.20884/1.jutif.2022.3.1.145
Rijanandi, T., Silvia, A., Abillah Safna, B., & Dias Ramadhani, R. (2023). Implementation of Encrypt National ID Card in Sinovi Application Use Waterfall Methodology. *RIGGS: Journal of Artificial Intelligence and Digital Business*, *1*(2), 11–18. https://doi.org/10.31004/riggs.v1i2.15
Rumetna, M. S., Lina, T. N., Rajagukguk, I. S., & Pormes, F. S. (2022). Payroll Information System Design Using Waterfall Method. *International Journal of Advances in Data and Information Systems*, *3*(1), 1–10. https://doi.org/10.25008/ijadis.v3i1.1227
Sanmocte, E. M. T., & Costales, J. A. (2025). Exploring Effectiveness in Software Development : A Comparative Review of System Analysis and Design Methodologies. *International Journal of Computer Theory and Engineering*, *17*(1), 37. https://doi.org/10.7763/IJCTE.2025.V17.1367
Sulfikar Sallu, Yhonanda Harsono, O. F. (2024). Implementation of Waterfall Method in Model Development to Improve Learning Quality of Computer Network Courses. *Jurnal Teknologi Pendidikan*, *25*(December 2023), 496–513. https://doi.org/10.21009/JTP2001.6
Yulianti, Adelia Dewi, Iqbal Ardiansyah, Moch. Agung Maulana, S. M. (2022). Perancangan Sistem Informasi Akademik Menggunakan Metode Waterfall Berbasis Web di SD Muhammadiyah 3 Ciledug. *Jurnal Informatika Universitas Pamulang*, *7*(3), 685–697. https://doi.org/10.32493/informatika.v7i3.21382