DEVELOPMENT OF A CLOUD-BASED SCHOOL MANAGEMENT INFORMATION SYSTEM USING GOOGLE APP SCRIPT AT SD MUHAMMADIYAH KADISOKA

Conduct by:

Name : Helga Parama Zhafran

Student ID : 21523044

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2026

SUPERVISOR ENDORSEMENT PAGE

Development of a Cloud-Based School Management Information System Using Google App Script at SD Muhammadiyah Kadisoka

THESIS

Conduct By:

Name : Helga Parama Zhafran

Student ID : 21523044

Yogyakarta, January 9th, 2026

Advisor,

(Dr. Syarif Hidayat, S.Kom., M.I.T.)

EXAMINER ENDORSEMENT PAGE

DEVELOPMENT OF A CLOUD-BASED SCHOOL MANAGEMENT INFORMATION SYSTEM USING GOOGLE APP SCRIPT AT SD MUHAMMADIYAH KADISOKA

THESIS

Has been defended in front of the examiners as one of the requirements to obtain a Bachelor of Informatics degree from the Undergraduate Program in Informatics at the Faculty of Industrial Technology, Universitas Islam Indonesia

Yogyakarta, January 9th, 2026

Chair

Dr. Syarif Hidayat, S.Kom., M.I.T. _____

Examiner 1

Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. _____

Examiner 2

Ari Sujarwo, S.Kom., M.I.T. _____

Acknowledged by,

Head of Undergraduate Program in Informatics

Faculty of Industrial Technology

Universitas Islam Indonesia

(Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

AUTHENTICITY STATEMENT

iv

AUTHENTICITY STATEMENT

The undersigned:

Name : Helga Parama Zhafran

Student ID : 21523044

Final project with title:

DEVELOPMENT OF A CLOUD-BASED SCHOOL MANAGEMENT INFORMATION SYSTEM USING GOOGLE APP SCRIPT AT SD MUHAMMADIYAH KADISOKA

Stating that all components and contents in this final project are my own work. If in the future it is proven that some parts of this work are not my own work, the final project submitted as my own work is ready to be withdrawn and ready to bear any risks and consequences.

Thus this statement letter is made, hopefully it can be used properly.

Yogyakarta, January 9th, 2026

(Helga Parama Zhafran)

DEDICATION

This thesis is dedicated to Allah SWT for His guidance and enduring mercy. To my beloved parents, thank you for your unwavering support, sincere prayers, and practical help. To myself: for resilience and perseverance. I am also grateful to all those who offered encouragement, time, and assistance, and to my friends who accompanied me and taught me invaluable lessons during my studies.

MOTTO

The challenge is to stay cool enough to handle the pressure in the moment so that you can succeed in the future. – Jurgen Klopp

FOREWORD

Assalamu’alaikum warahmatullahi wabarakatuh.

Alhamdulillahi Rabbil ‘Alamin, all praise be to Allah SWT, who has bestowed upon the author strength, patience, health, and guidance throughout the completion of this undergraduate thesis, entitled “Development of a Cloud-Based Religious School Management Information System to Facilitate School Data Management.” Salawat and Salam are also extended to the Prophet Muhammad SAW, his family, companions, and followers until the end of time.

This Final Project is submitted as a partial requirement to obtain a Bachelor's Degree in Informatics. Throughout the preparation and completion of this work, the author encountered several non-academic challenges, including time constraints, limited resources, and balancing academic responsibilities. However, with continuous support, encouragement, and the will of Allah SWT, this research was successfully completed.

1. The author’s family, especially the parents Nurman Kurniawan and Yuliana, for their unwavering support, prayers, guidance, and continuous provision of material and emotional strength throughout this journey.
2. Dr. Syarif Hidayat, S.Kom., M.I.T., the thesis supervisor, for the invaluable guidance, constructive feedback, and continuous encouragement given during the research and writing process.
3. All friends who accompanied the author throughout the research and writing stages, offering motivation, support, and helpful discussions.
4. Fadilla Nur Amalia, whose brilliance, encouragement, and steadfast support from the beginning have been deeply meaningful to the author’s academic journey.
5. SD Muhammadiyah Kadisoka — Pak Suharyanto, S.Pd. as Headmaster, Bu Tasya as Teacher, and Pak Arif as Multimedia Teacher/Staff — for their warm welcome, cooperation, and constructive feedback that greatly assisted the development of this research.
6. Liverpool FC, the author’s favorite football club, whose motto "You’ll Never Walk Alone" has served as a source of motivation and resilience during difficult and uplifting moments alike.
7. Ir. Irving Vitra Paputungan, S.T., M.Sc., Ph.D., the academic advisor, for providing continuous guidance, academic direction, and insight throughout the author’s undergraduate studies.

8. Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng., Head of the Undergraduate Program in the Informatics Study Program, for fostering a supportive and constructive academic environment.

The author recognizes that this thesis is not without shortcomings and sincerely welcomes constructive criticism and suggestions for future improvements. It is hoped that this work may provide meaningful contributions to the development of information systems and serve as a reference for future research.

This thesis has been prepared as part of the requirements for obtaining a Bachelor's Degree in the Informatics International Undergraduate Program, Industrial Technology Faculty, Universitas Islam Indonesia.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Yogyakarta, January 9th, 2026

(Helga Parama Zhafran)

ABSTRACT

Many schools in Indonesia continue to face obstacles in adopting digital administration systems due to limited infrastructure, financial constraints, and the absence of technical personnel. Consequently, daily operations often rely on fragmented paper records, offline spreadsheets, or government reporting platforms that do not support routine academic workflows. To address these limitations, this research designs and validates a cloud-based School Management Information System specifically for SD Muhammadiyah Kadisoka.

The system uses a serverless architecture built on the Google ecosystem (Google Apps Script, Google Sheets, and Google Drive) to ensure zero-cost execution and minimal maintenance. Key features include role-based user management, automatic grade conversion, automatic unique-ID generation, document upload and archiving, dashboard visualizations, and automated WhatsApp notifications via the Wablas API which remove repetitive manual steps and reduce opportunities for human error. The study documents a specific platform limitation: Google Apps Script and Sheets impose execution time and concurrency quotas that affect long-running or highly concurrent operations; these constraints are technical platform limits (not limitations of the WhatsApp gateway) and are discussed with concrete mitigations.

System validation employed black-box testing and the System Usability Scale (SUS) with 13 stakeholders, including administrators, teachers, and student guardians. Functional testing confirmed a 100% success rate across all specifications. The SUS evaluation yielded an average score of 74.77, indicating "Good" usability. Notably, internal school staff reported higher satisfaction (81.6) compared to guardians (72.75), highlighting the effectiveness of the system for administrative tasks while suggesting a need for user guidance for parents. The findings demonstrate that a serverless Google-based solution delivers a reliable, scalable, and affordable management system, providing a practical model for resource-constrained schools to streamline administration without requiring local servers.

Keywords: school management information system, google apps script, serverless architecture, waterfall model, system usability scale, educational technology.

GLOSSARY

API	a set of rules and tools that allows different software applications to communicate with each other.
Black-box Testing	a software testing method that evaluates the functionality of an application without examining its internal code structure.
Cloud Computing	a model for delivering computing services—including servers, storage, databases, networking, and software—over the internet.
CRUD Operations	the four basic functions of persistent storage: Create, Read, Update, and Delete.
Dapodik	the Indonesian Ministry of Education's Primary Data for Education system.
ERD	a visual representation of the data model, showing the relationships between key entities in the system.
Google Apps Script	a cloud-based JavaScript platform developed by Google.
Google Workspace	a collection of cloud computing, productivity, and collaboration tools developed by Google, including Google Sheets, Drive, and Apps Script.
MIS	an integrated, user-machine system designed to provide information to support operations, management, and decision-making within an organization.
RPP	a mandatory teaching document prepared by teachers in Indonesia detailing the learning objectives, materials, methods, and assessment for a lesson or unit.
SUS	a reliable and widely used questionnaire for measuring the perceived usability of a system. It consists of ten items rated on a five-point scale.
Wablas	a third-party commercial API service that provides a gateway to send automated messages via WhatsApp.
Waterfall	a method of software development.

TABLE OF CONTENTS

LIST OF TABEL

LIST OF FIGURES

CHAPTER I
INTRODUCTION

1.1 Background

The global digital transformation has made information technology integral to educational administration, a trend the Indonesian government actively supports through platforms such as Data Pokok Pendidikan (Dapodik) to improve efficiency and transparency (Rodiyah, Rustianingsih, & Solicha, 2024). However, this progress is uneven: many religious-based elementary schools (madrasah/sekolah Islam) operate with limited technical and financial resources, which restricts their ability to adopt and sustain conventional information systems (S, Ernawati, Saputra, & Kurniawan, 2024).

SD Muhammadiyah Kadisoka illustrates this disparity. Preliminary observations and interviews with the Student Affairs Department, the principal, and administrative staff indicate that the school depends mainly on Dapodik for external government reporting but lacks a unified internal platform for daily administration. Consequently, routine tasks, such as lesson-plan submission (RPP), grade recording, schedule dissemination, and parent communication, are managed through fragmented paper archives and offline spreadsheet files. These manual processes are labor intensive, prone to human error, and create data redundancy and inconsistency between internal records and government reporting.

The school also faces infrastructure and maintenance challenges that make on-premise server-based solutions impractical. Hardware acquisition and upkeep, electricity costs, and the absence of dedicated IT personnel present significant barriers. In this context, cloud computing, specifically a serverless architecture, provides a viable alternative by eliminating the need for local servers while offering broad accessibility to administrators, teachers, students, and guardians (Shanganlall, 2024).

This research uses a case-based engineering approach to design and validate a cloud-based School Management Information System (MIS) tailored to the school's operational context. The proposed solution leverages the Google Workspace ecosystem (Google Apps Script, Google Sheets, and Google Drive) as a zero-cost, low-maintenance backend and integrates WhatsApp automation via the Wablas API to strengthen school–parent communication. Development follows a structured Waterfall methodology, beginning with requirements elicitation and proceeding through design, implementation, and validation.

1.2 Problem Formulation

Based on the background outlined above, the problems identified at SD Muhammadiyah Kadisoka are formulated as follows:

a. There is no integrated digital platform that enables teachers to upload teaching documents (e.g., lesson plans) or to input grades digitally.
b. Students and guardians do not have a dedicated interface to access academic schedules and real-time grade information, relying instead on manual inquiries.
c. The institution cannot support conventional server-based software solutions because of limited financial and technical resources for hardware maintenance and IT personnel.
d. Administrative workflows depend on scattered offline spreadsheets and paper records rather than a centralized database, causing data redundancy and making internal report generation inefficient.

1.3 Research Objectives

The primary objective of this research is to design, develop, and validate a cloud-based School Management Information System (MIS) that directly addresses the operational limitations identified at SD Muhammadiyah Kadisoka. The specific technical objectives are:

a. To develop a zero-cost, serverless web application using Google Apps Script (backend), Google Sheets (database), and Google Drive (file storage) that eliminates the need for local server hardware and ongoing server maintenance.
b. To implement role-based functionality for Administrators, Teachers, and Students/Guardians, including:
    1. Administrator: full CRUD (Create, Read, Update, Delete) operations for master data and school configuration; broadcast messaging controls.
    2. Teacher: profile management, digital upload of lesson plans, and grade input.
    3. Student/Guardian: access to personal profiles, academic grades, and class schedules, plus limited profile update capability.
c. To integrate a WhatsApp notification gateway using the Wablas API to automate school–parent communications leveraging a widely used messaging channel.
d. To validate the system’s functionality via Black-box testing and evaluate usability using the System Usability Scale (SUS), targeting an average score that meets or exceeds commonly accepted thresholds for acceptable usability.

1.4 Benefits of the Research

The development and deployment of the proposed MIS are expected to provide the following benefits:

a. This research produces a low-cost and easy-to-maintain MIS prototype that can be implemented by religious schools or similar institutions with limited financial and technical resources.
b. This research provides a validated case study demonstrating that Google Sheets can function as a viable serverless database for simple web applications.
c. This research introduces a real-world case study on integrating the Wablas API for WhatsApp notifications, which enhances communication between schools and parents.
d. This research demonstrates the suitability of the Waterfall methodology for projects with clear, fixed requirements in an educational context, contributing to the body of knowledge on software development practices for non-profit or community-based organizations.
e. This research demonstrates improved operational efficiency through process automation, where routine administrative tasks are streamlined by replacing manual calculations and communications with automated workflows.
f. The system eliminates the need for physical visits or manual inquiries regarding schedules and grades. Guardians gain immediate, self-service access to academic data, resolving the transparency gap inherent in the previous manual system.

1.5 Scope and Limitations

To keep the research focused and feasible, the following scope and limitations are defined:

a. This research is a case study of SD Muhammadiyah Kadisoka and is tailored to its operational context. Findings may inform similar institutions but are not automatically generalizable without adaptation.
b. The system is implemented using Google Apps Script, Google Sheets, and Google Drive. It is a standalone MIS prototype and does not constitute a custom PHP/SQL on-premise solution.
c. The system operates independently and is not integrated with the national Dapodik database or other external governmental platforms; any synchronization would require separate mechanisms.

d. The system focuses on core academic and administrative modules (student and teacher data, schedules, grade management, document uploads, and WhatsApp broadcasting). It does not include advanced modules such as financial accounting, payroll.
e. The database layer uses Google Sheets, which is not designed for high-frequency transactional locking. The system is optimized for low-to-moderate concurrent usage (approximately 30 simultaneous executions, consistent with Google Apps Script quotas). Heavy simultaneous write operations may result in errors or increased latency due to spreadsheet write-locking.
f. The backend logic is bound by Google Apps Script's maximum runtime quota (6 minutes per execution). Timeouts are most likely to occur during backend-intensive CRUD operations.
g. Google Sheets does not enforce strict data types or foreign key constraints at the database level. Data integrity is maintained entirely through the application logic (backend validation) rather than database-level enforcement.

1.6 Research Structure

This research is structured into five distinct chapters to systematically achieve the stated objectives. The methodology follows a sequential process, beginning with problem identification and concluding with evaluation and recommendations.

a. CHAPTER I: INTRODUCTION This chapter establishes the foundation of the research. It details the background context, defines the core problem, presents the research questions, outlines the scope and limitations, and states the objectives and contributions of the study.
b. CHAPTER II: LITERATURE REVIEW This chapter reviews existing research and relevant theories. It examines the specific challenges faced by religious schools, compares software development methodologies (Waterfall, Agile, etc.), and evaluates potential cloud-based data storage solutions (Google Sheets, Cloud SQL, etc.) to justify the technical choices for this project.
c. CHAPTER III: METHODOLOGY This chapter provides a detailed blueprint of the development process. It explains the application of the Waterfall model, presents the system architecture through solution and use case diagrams, outlines the functional and data requirements gathered through qualitative methods, and describes the testing plan based on the System Usability Scale (SUS).

d. CHAPTER IV: SYSTEM IMPLEMENTATION AND TESTING This chapter will detail the execution phase of the project. It will describe the technical implementation of the web-based interface and its integration with the Google Sheets backend. It will also present and analyze the results obtained from the SUS usability testing conducted with school staff.

e. CHAPTER V: CONCLUSION AND RECOMMENDATIONS This final chapter will summarize the key findings of the research, drawing conclusions based on the project's outcomes in relation to the initial objectives. It will also provide recommendations for future enhancements to the system and suggest avenues for further research.

CHAPTER II
LITERATURE REVIEW

2.1 Management Information Systems in Education

A Management Information System (MIS) is an integrated user-machine system for providing information to support operations, management, and decision-making functions in an organization (Gupron, Yandi, Suprapto, & Sadewa, 2024). In the educational context, an MIS is designed to manage student data, academic grades, scheduling, and personnel records to enhance administrative efficiency (Mandei, Siang, & Mamahit, 2025).

In Indonesia, the government has implemented Data Pokok Pendidikan (Dapodik) as a national-scale data collection system. However, studies indicate a "digital divide" where schools fulfill top-down reporting requirements for Dapodik but lack internal systems for daily operations (Kasbijanto, Mohammad Syahidul Haq, Amalia, & Hazin, 2025). For religious-based institutions like SD Muhammadiyah Kadisoka, reliance solely on Dapodik creates operational gaps, as the platform does not provide interfaces for parents to check grades or for teachers to upload lesson plans (Rencana Pelaksanaan Pembelajaran or RPP) digitally. Consequently, the development of an internal, school-specific MIS is necessary to bridge the gap between national reporting and daily administrative needs.

2.2 Cloud Computing and Serverless Architecture

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Moss, 2016). This model is characterized by essential attributes: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service (Liu & Zeng, 2024).

A serverless architecture is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers (Ghorbian & Ghobaei-Arani, 2025). The application developer does not see, manage, or interact with the underlying servers; they simply deploy code (Gandhi, Gore, Nimbarte, & Abraham, 2018). This is distinct from traditional server-based models (on-premise or Infrastructure-as-a-Service) which require

ongoing management, scaling, and maintenance of server instances (Veuvolu, Suryadevar, Vignesh, & Avthu, 2023).

In educational institutions where system sustainability and ease of operation are critical, serverless computing has increasingly been discussed as a viable alternative to on-premise and Infrastructure-as-a-Service (IaaS) models. Prior research emphasizes that minimizing server administration tasks and recurring infrastructure management lowers long-term dependency on specialized technical personnel while supporting continuous system availability.

2.3 Review of Core Technologies

The selection of technological components plays a critical role in determining the sustainability and usability of educational information systems. Prior research emphasizes that technology adoption in small educational institutions should prioritize simplicity, low maintenance overhead, and alignment with users' technical capabilities rather than maximum performance or scalability (Abdugulova, 2017). Accordingly, several studies have explored lightweight platforms and managed cloud services as alternatives to conventional enterprise software stacks.

This section reviews the core technologies frequently discussed in the literature for developing low-cost, cloud-based information systems, with a focus on platforms that support rapid development, minimal infrastructure management, and integration with widely used productivity tools.

2.3.1 Google Apps Script

Google Apps Script (GAS) is a rapid application development platform that makes it fast and easy to create business applications that integrate with Google Workspace ("Google Apps Script Overview," n.d.). It is built on JavaScript (V8 engine) and runs on Google's cloud infrastructure. Google Apps Script is suitable for lightweight web applications in resource-constrained environments due to its native integration with Google Workspace and serverless execution model (Ferrell, 2018). For the studied school where cost and technical simplicity are paramount, it offers an unparalleled advantage by providing direct, API-free access to Sheets and Drive and thus eliminating the backend complexity and hosting expenses that pose significant barriers with Node.js or Flask, as shown in Table 2.1.

Table 2.1 Comparison of Backend Runtime Environments

Feature	Google Apps Script	Node.js / Python (Flask)
Environment & Cost	Serverless, fully managed by Google. Zero hosting cost.	Requires a server/hosting platform (e.g., AWS, Heroku), incurring subscription costs (Bulla & Rao, 2019).
Primary Use Case	Automating and extending Google Workspace applications.	General-purpose web development.
Integration with Google Services	Native and seamless (e.g., SpreadsheetApp, DriveApp).	Requires external libraries (e.g., googleapis) and complex OAuth 2.0 setup (Jangid, 2025).
Development Overhead	Minimal setup; development starts immediately within the browser.	Requires local environment setup, dependency management, and deployment configuration.

2.3.2 Google Sheets as a Database

Google Sheets is primarily designed as a spreadsheet application, but it can function effectively as a flat-file database for small-scale applications (admin, 2021). In this system, Google Sheets are structured to emulate relational database tables, where columns represent data attributes (such as NISN, Nama, and Kelas in the student dataset), rows correspond to individual records, and separate sheet tabs serve as logical tables (for example, sheet for Student, sheet for Teacher, sheet for save student grade, and schedule). In religious-based elementary schools managing a few hundred students and dozens of teachers, the scalability constraints of Google Sheets are typically non-critical, while its zero-maintenance characteristic offers a substantial advantage in environments with limited technical and financial capacity (Sander, 2025).

The use of spreadsheet-based storage as an alternative to conventional database management systems has been examined in several studies addressing educational and administrative information systems with moderate data volume and controlled access patterns (Bendre, Venkataraman, Zhou, Chang, & Parameswaran, 2018). In such implementations, spreadsheet platforms are structured to represent tabular datasets, where rows correspond to records and columns define data attributes (Bendre et al., 2015).

The literature consistently identifies limitations associated with spreadsheet-based storage, including weak schema enforcement, limited transaction control, and constrained scalability compared to relational database systems. These limitations are reported to become critical primarily in environments characterized by high concurrency, complex relational queries, or mission-critical data processing.

Conversely, studies indicate that for academic administrative systems with predictable data growth and structured workflows, spreadsheet-backed storage can operate reliably when supported by access control mechanisms and standardized data conventions (Thorne & Hancock, 2019). However, a critical distinction noted in database theory is the issue of write contention. Unlike SQL databases which support row-level locking, Google Sheets generally locks the file during write operations to maintain integrity. This characteristic makes the system susceptible to latency or data collision if dozens of users attempt to write data (e.g., saving grades) at the exact same millisecond (Murphy, 2008). As a result, Google Sheets is increasingly framed in the literature as a pragmatic data storage solution for non-critical, low-concurrency academic administration, rather than as a substitute for enterprise-grade transactional database platforms. A comparison of the data storage solutions used in this study is shown in Table 2.2.

Table 2.2 Comparison of Data Storage Solutions

Feature	**Google Sheets**	**Relational Database (e.g., MySQL, PostgreSQL, Cloud SQL)**
Cost	Free with generous usage quotas.	Requires server, hosting, or subscription fees.
Ease of Use	Very high; familiar spreadsheet interface suitable for non-technical users.	Low to moderate; requires SQL knowledge.
Maintenance	None; fully managed as a SaaS platform.	High; requires database administration and maintenance.
Setup	Instant; sheets can be created within seconds.	Requires installation, configuration, and setup.
Data Integrity	Low; no enforced schema or relational constraints.	Very high; supports strict schema and ACID transactions.
Scalability	Limited; up to approximately 5 million cells.	High; designed for large-scale data handling.
User Interface	Visual and grid-based, user-friendly.	Query-based, less intuitive for non-technical users.

2.3.3 Google Drive as a File Storage

Google Drive is employed to store uploaded documents such as lesson plans and student photographs, and its free tier adequately meets the school's storage requirements (Andi Bulkis Magfirah Mannong, 2024). Cloud-based file storage services are commonly differentiated between developer-oriented object storage platforms and user-oriented document management systems (Weis, 2024). Prior studies note that object storage services such as Amazon S3 provide high scalability and fine-grained access control but are primarily designed for

programmatic interaction through APIs and developer-centric workflows (Surahman, 2023). A comparison of these storage options is presented in Table 2.3.

In contrast, user-oriented cloud storage platforms such as Google Drive emphasize direct usability, visual file organization, and accessibility for non-technical administrative users. Research on educational information systems suggests that such platforms reduce operational friction and training requirements while remaining sufficient for document-centric academic workflows, including instructional materials and administrative records (Pollard, 2024).

Table 2.3 Comparison of File Storage Solutions

Feature	Google Drive	Cloud Object Storage (e.g., Amazon S3)
Pricing Model	15 GB free tier; simple, fixed-price plans.	Pay-as-you-go for storage, requests, and data transfer.
User Experience	Excellent; file-and-folder interface universally understood.	Technical; designed for developers via API or complex console.
Integration with Apps Script	Native via DriveApp service.	Requires external API calls and authentication handling.

2.3.4 Wablas API (Communication Gateway)

The system incorporates the Wablas API as a communication gateway to facilitate automated WhatsApp messaging between the school and parents. Wablas is a third-party Application Programming Interface (API) that enables web applications to interact with the WhatsApp network, allowing messages to be sent programmatically. This integration addresses communication gaps by replacing manual methods such as paper notices or SMS with automated notifications, including registration confirmations and broadcast messages. Given that WhatsApp is the dominant communication platform in Indonesia, embedding the Wablas API within the Management Information System leverages an existing and widely adopted channel, thereby ensuring higher message delivery rates, improved parental engagement, and more efficient school–parent communication (Siskawati, Masturi, Hidayati, Sary, & Nisoh, 2025).

2.4 Software Development Methodology

The selection of an appropriate software development methodology significantly influences project structure, control, and outcomes. In the literature, software development approaches are commonly categorized into plan-driven models, such as the Waterfall model, and iterative models, such as agile frameworks (e.g., Scrum). This research adopts the

Waterfall model based on a comparative evaluation of these approaches in relation to the specific context of developing a Management Information System (MIS) for SD Muhammadiyah Kadisoka.

The Waterfall model follows a linear and sequential lifecycle consisting of clearly defined phases, including requirements analysis, system design, implementation, testing, and maintenance (Bassil, 2012). Each phase is completed and documented before proceeding to the next, emphasizing upfront planning and formal documentation. In contrast, agile methodologies emphasize iterative development through short cycles, allowing requirements to evolve continuously and relying on frequent stakeholder collaboration (University of Novi Sad - Faculty of Economics, Segedinski put 9-11, 24000 Subotica, Matković, Tumbas, & University of Novi Sad - Faculty of Economics, Segedinski put 9-11, 24000 Subotica, 2010).

2.5 System Testing and Evaluation Methods

To ensure the system's quality and acceptance, two standard evaluation methods are referenced in this study: Black-Box Testing and the System Usability Scale (SUS).

2.5.1 Black-Box Testing

Black-box testing is a software testing method that examines the functionality of an application without peering into its internal structures or workings (Frayudha, Pande, & Juwita, 2024). The primary focus is on the inputs and expected outputs. This method is theoretically grounded in ensuring that every specific function (e.g., "Save Grade," "Upload RPP," "Send Broadcast") operates according to the user requirements specification (Sasmito & Mutasodirin, 2023). It is the most common testing method found in similar academic research due to its effectiveness in validating functional correctness.

2.5.2 System Usability Scale (SUS)

The System Usability Scale (SUS) is a reliable and popular tool for measuring the usability of a system (Clark et al., 2021). It consists of a 10-item questionnaire with five response options for respondents, ranging from "Strongly agree" to "Strongly disagree." (“System Usability Scale - an Overview | ScienceDirect Topics,” n.d.).

Theoretically, SUS yields a single number representing a composite measure of the overall usability of the system. Scores are calculated as follows:

- Items 1, 3, 5, 7, and 9 are the positive items (Score = User Response - 1).

- Items 2, 4, 6, 8, and 10 are the negative items (Score = 5 - User Response).
- The sum of the scores is multiplied by 2.5 to obtain the overall value (0–100).

A SUS score above 68 is considered "Average," while scores above 80 are considered "Excellent.". This metric provides a quantitative baseline to validate whether the developed MIS is user-friendly for non-technical staff and guardians.

2.6 Related Works and Research Gap

To position this research within the broader academic context, a review of previous studies regarding School Information Systems (SIM) was conducted. Table 2.4 presents a summary of relevant studies, analyzing their methods, and results.

Table 2.4 Review of Prior Research

No	Title	Author / Year	Methodology	Result
1	Implementasi Google App Script untuk Data Entry pada Master Data	(Rosanti & Swalagan ata, 2024)	Website development with HTML/CSS/JS and Google Apps Script; uses SDLC with Waterfall / FDD for design.	Produces a data-input website backed by Google Sheets that speeds up filtering and updating master data.
2	Implementation of Google App Script in Cloud-Based Data Search Application	(Insan Asry, 2022)	Experimental development of cloud-based search using Google Apps Script and Google Sheets.	Demonstrates that Apps Script can retrieve and display spreadsheet data on a web interface efficiently.
3	Sistem Informasi Kehadiran Siswa Berbasis Web Menggunakan API WhatsApp di SMK Negeri 1 Al Mubarkeya Aceh Besar	(Inda Fathya, 2025)	Web-based attendance information system using PHP and MySQL; SDLC Waterfall from analysis through implementation and testing.	System sends automatic WhatsApp notifications to parents when students are absent without explanation, improving timeliness of attendance information.
4	Sistem Informasi Monitoring Kegiatan Absensi Siswa Dengan Notifikasi WhatsApp	(Dedi Jubaedi et al., 2023)	Web-based monitoring system using WhatsApp notifications; developed with the Waterfall method (analysis, design, implementation, testing).	Generates real-time WhatsApp alerts so teachers and parents can monitor attendance more effectively and reduce absenteeism.
5	Design of Web-based Report Card Processing Information System at MTS Negeri 4 Kebumen	(Hasim Iswanto, 2024)	Web-based report-card system using PHP–MySQL; SDLC Waterfall with clearly defined stages and functional testing.	System successfully manages K13 report-card data; functional suitability results fall in the "excellent" category and improve efficiency of grade processing.
6	Whatsapp Integrated Information System on Student Attendance Management: Sistem Informasi Terintegrasi Whatsapp pada Manajemen Absensi Siswa	(Pratama & Hindarto, 2025)	Web-based report-card management system; structured development process aligned with Waterfall-type SDLC (analysis to testing).	Enables teachers and staff to manage student grades from anywhere and reduces risk of data loss compared to manual processing.

A review of prior studies on school information systems reveals several dominant trends. Many existing systems employ conventional server-based architectures using technologies such as PHP and MySQL, which require continuous hosting, maintenance, and technical administration. While these systems demonstrate functional effectiveness, their infrastructure requirements often exceed the operational capacity of small religious-based schools with limited financial and human resources.

Prior research involving Google Apps Script and Google Sheets predominantly focuses on isolated automation tasks, such as data entry or search applications, rather than the development of integrated, multi-role management information systems. These studies validate the technical feasibility of spreadsheet-backed applications but do not address comprehensive academic workflows involving administrators, teachers, students, and guardians within a single platform.

Studies that integrate WhatsApp-based communication primarily emphasize attendance monitoring or notification delivery, with limited exploration of deeper academic integration such as grading, scheduling, and document submission. Furthermore, these systems are commonly implemented using traditional server-based backends rather than fully managed, serverless environments.

Based on these findings, a research gap can be identified in the lack of documented case studies that examine the development and evaluation of a fully serverless, zero-cost school management information system that integrates academic administration, teacher workflows, and parent communication within a unified platform. This study addresses that gap by designing and empirically evaluating a cloud-based MIS built entirely on managed Google services and a third-party WhatsApp gateway, with usability and functional reliability assessed through black-box testing and the System Usability Scale.

CHAPTER III
METHODOLOGY

This chapter details the systematic methodology employed to design, develop, and validate the cloud-based School Management Information System (MIS) for SD Muhammadiyah Kadisoka. The primary objective is to construct a functional, zero-cost, and user-friendly system that directly addresses the school's operational constraints, as identified in Chapter I. To ensure a structured and manageable development process within the context of a case study with fixed requirements, the Waterfall software development lifecycle model was adopted. This chapter elaborates on the sequential phases of the methodology: Requirements Analysis, System Design, Implementation, and Testing & Evaluation. The workflow, illustrated in Figure 3.1, provides a clear roadmap from problem identification to solution validation.

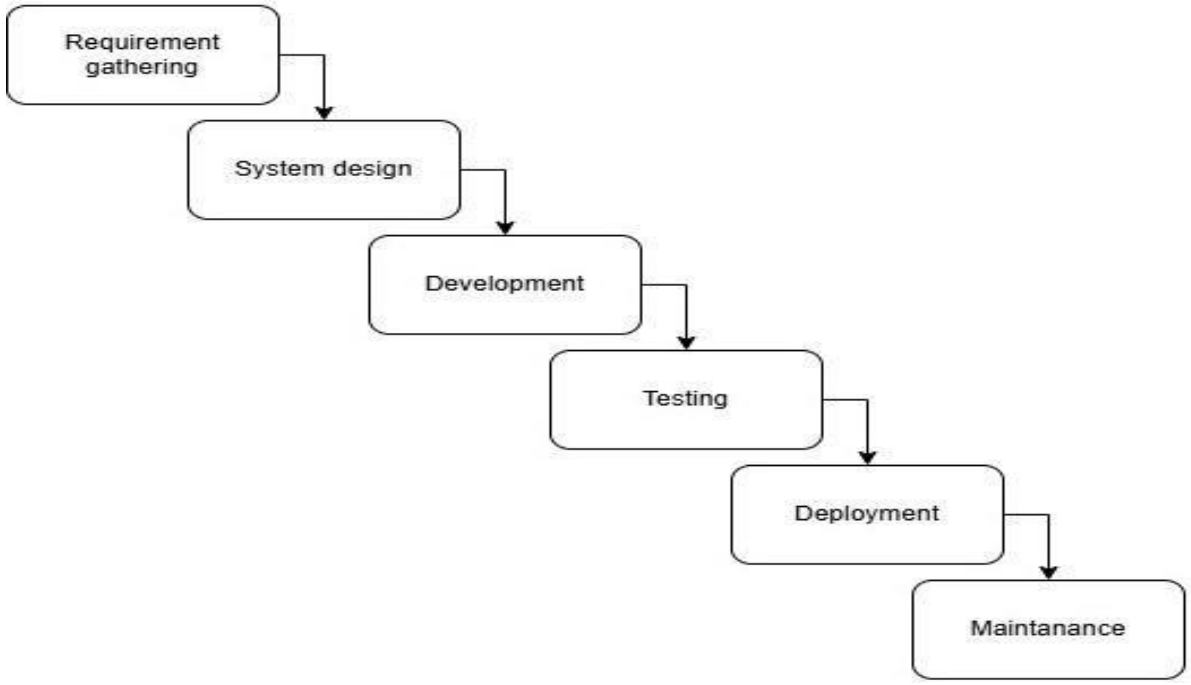

Figure 3.1 Waterfall Methodology Workflow

3.1 Requirement Analysis

A qualitative, case-study approach was used to gather precise requirements, ensuring the system aligns with the specific daily operations and constraints of SD Muhammadiyah Kadisoka.

3.1.1 Data Collection Methods

To ensure the system addresses the specific operational needs of SD Muhammadiyah Kadisoka, data collection was conducted through two primary methods:

a. Observation, Preliminary observations identified that while the school complies with national data reporting via Dapodik, this platform is accessible only to administrators. There is no internal ecosystem for students or guardians to check grades and schedules in real-time, nor for teachers to digitally upload "Teaching Plans" (RPP), which currently requires physical submission. Furthermore, it was observed that the school lacks the budget and technical personnel to maintain a local on-premise server.

b. Interviews, Interviews were conducted with key stakeholders to define functional requirements and gather essential institutional data:

    1. 5 November 2025: An interview was conducted with a teacher who also serves in the Student Affairs Department. This session identified the need for specific features such as a student profile portal, digital document uploads for teachers, grade management, and a WhatsApp (WA) broadcast feature. Crucially, this interview provided the specific population data required for the system's database sizing: the school serves approximately 500 students and employs 56 teachers, with the student body distributed across 18 classes divided into three clusters each (e.g., 1a, 1b, 1c, 2a, etc.).
    2. 19 November 2025: A strategic interview was held with the School Principal and two administrative staff members. This session confirmed the core operational constraint: the school cannot rely on server-based solutions due to high maintenance and electricity costs. Consequently, the requirement was set for a cloud-native, serverless solution.

3.1.2 Identified Requirements

The analysis revealed the school’s sole reliance on the national Dapodik platform for external reporting and the complete absence of an integrated internal digital platform. The requirements obtained are:

a. The system must operate without local servers or hosting fees due to zero budget for IT infrastructure and maintenance.

b. The system must provide distinct interfaces and functionalities for three primary user roles:

1. Administrator: Capabilities for managing master data (students, teachers, classes) and send broadcast messages.
2. Teacher: Ability to digitally upload teaching documents (e.g., lesson plans) and input/update student grades for assigned classes.
3. Student/Guardian: Read-only access to academic schedules and grades, with limited ability to update personal contact information.

c. Integration with a widely used messaging platform to automate notifications (e.g., new account information, announcements) to parents.

3.2 System Analysis and Database Design

3.2.1 Use Case Diagram

The system's core interactions are illustrated in Figure 3.2, which delineates the functional scope and responsibilities associated with each actor within the system.

a. Actor: Administrator. Manages all foundational data, including students, teachers, classes, and academic schedules. Can send broadcast messages to parents.
b. Actor: Teacher. Manages personal profile, uploads lesson plans, and inputs/updates grades for students in their assigned classes.
c. Actor: Student/Guardian. Views personal academic schedule and grades. Updates limited personal profile information.

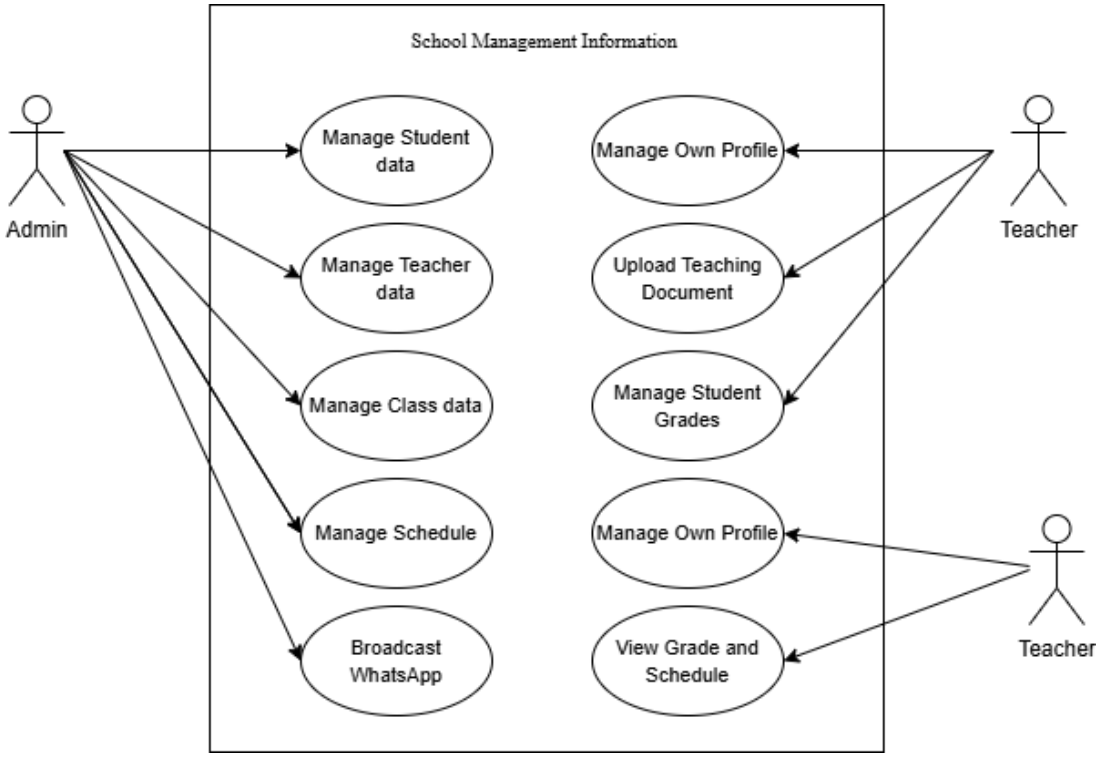

Figure 3.2 System Use Case Diagram

3.2.2 Activity Diagram

The activity diagrams, illustrated in Figure 3.3-Figure 3.12, specify the step-by-step workflows for the system's major functions. Each diagram maps user actions, system checks, data writes to Google Sheets/Drive, and external interactions (Wablas API), and highlights decision points and error-handling paths to ensure reproducibility and testability.

3.2.2.1 Admin Activity Diagram

The activity diagram illustrated in Figure 3.3 describes the administrator's workflow for managing student data. For adding a new student, the system validates required fields and data formats, generates a unique student ID, appends a new record to Google Sheet, uploads the student photo to the designated Google Drive folder, and stores the file link. A welcome notification is dispatched via the Wablas API only during this initial registration process. For editing student data, the system retrieves the existing record, applies validated updates, and overwrites the corresponding row in the sheet. For deleting data, the system removes the selected record after confirmation.

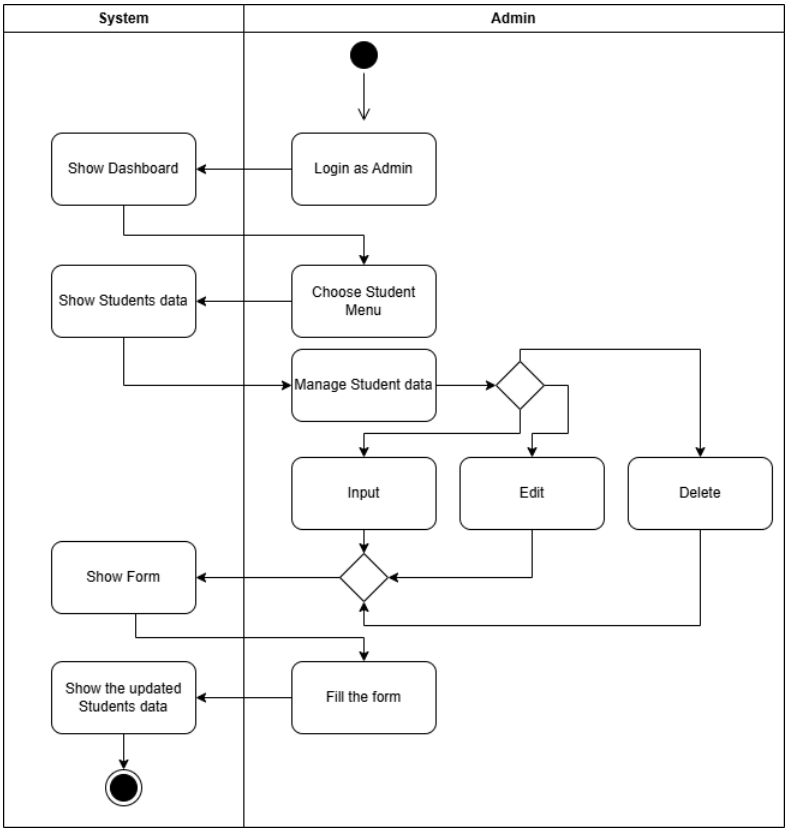

Figure 3.3 Activity Diagram for Admin Managing Student Data

The activity diagram illustrated in Figure 3.4 presents the administrator's workflow for managing teacher data, which follows a process similar to that used for managing student data. During the addition of a new teacher, the system validates the submitted information, generates a unique teacher identifier, stores the data in the corresponding Google Sheet, uploads any associated profile files to the designated Google Drive folder, and dispatches notification via the Wablas API as part of the initial registration process. For editing teacher data, the system retrieves the existing record, applies validated changes, and updates the relevant entry in the sheet. For deleting teacher data, the system removes the selected record after confirmation.

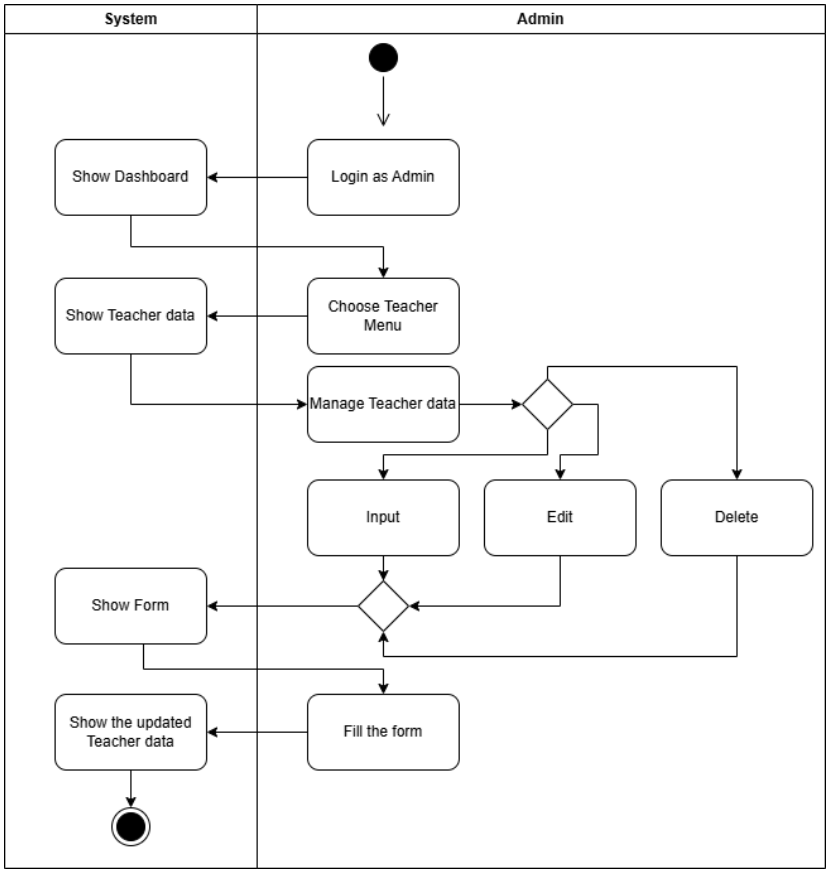

Figure 3.4 Activity Diagram for Admin Managing Teacher Data

The activity diagram illustrated in Figure 3.5 presents the workflow for building or updating a class schedule. The administrator selects the target class and day, assigns subjects and teachers to each period, and saves the schedule. The system validates inputs, resolves teacher–subject assignments, and writes the resulting timetable to the jadwalSiswa sheet. The diagram captures rollback or correction steps if conflicts are detected (e.g., teacher double-booking) and documents success/failure notifications to the administrator.

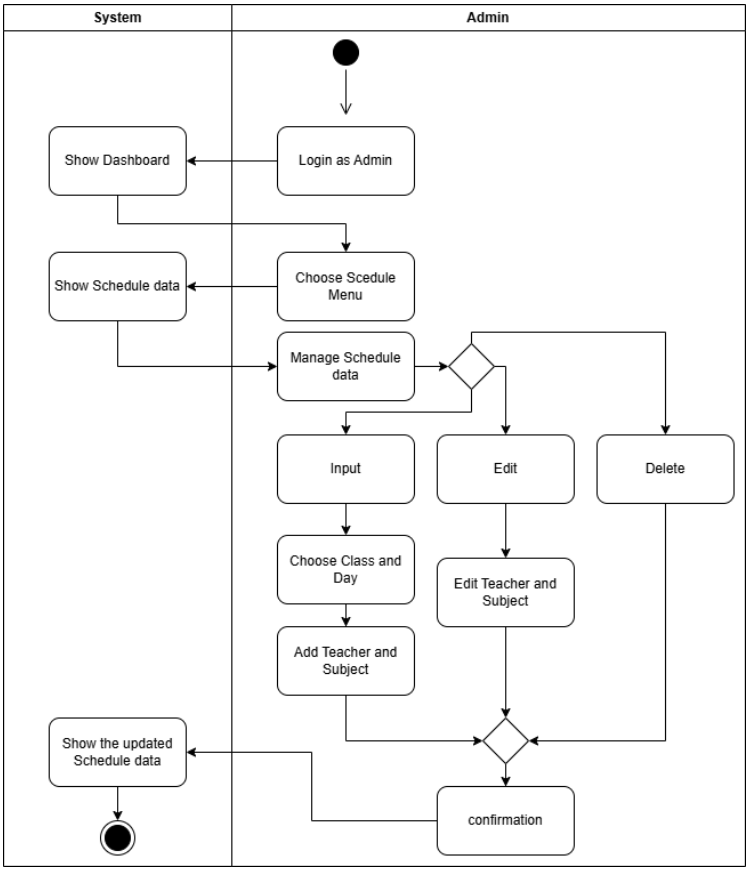

Figure 3.5 Activity Diagram for Admin Managing Schedule Data

The activity diagram illustrated in Figure 3.6 depicts the administrator's workflow for managing class data, including adding, updating, and deleting class records. When creating or editing a class, the administrator completes the class form and selects a wali kelas from a list of teachers; upon selection, the system automatically populates the corresponding teacher ID. The system then validates the input and saves or updates the class data in the relevant Google Sheet. For deletion, the selected class record is removed after confirmation.

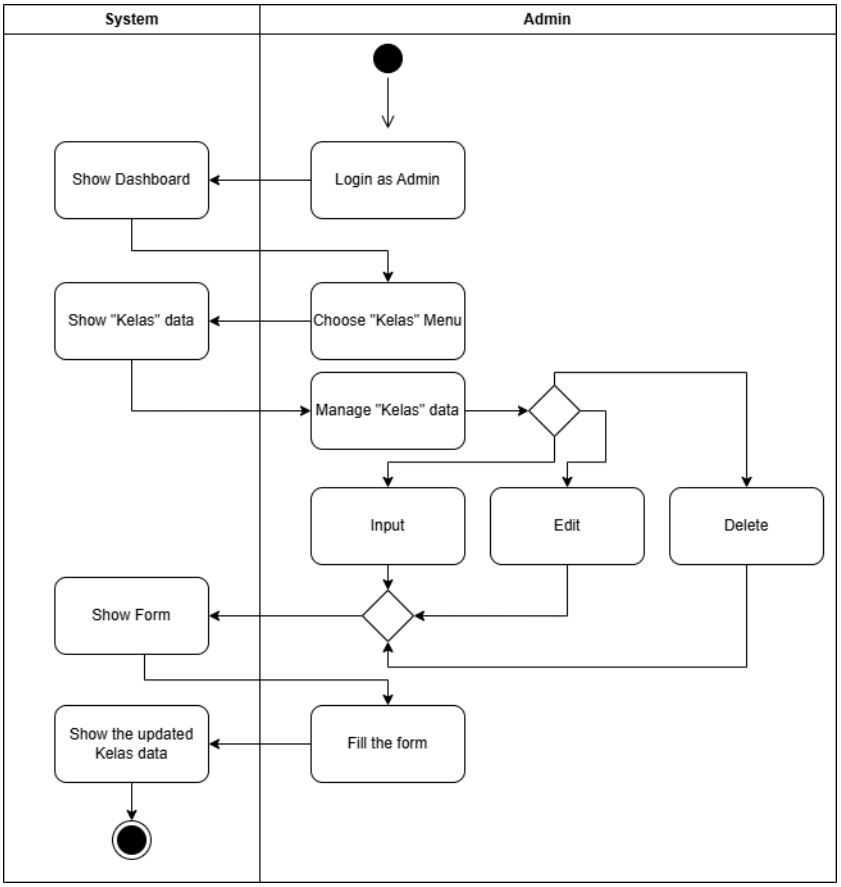

Figure 3.6 Activity Diagram for Admin Managing Class Data

As illustrated in Figure 3.7, after the administrator confirms the broadcast action, the system sends the WhatsApp message to each recipient sequentially via the Wablas API. For every delivery attempt, the system records the response status returned by the API. The final result page then displays a summary indicating which messages were successfully sent and which failed. Failed deliveries are logged to support administrative review and troubleshooting, such as identifying invalid phone numbers or transmission errors.

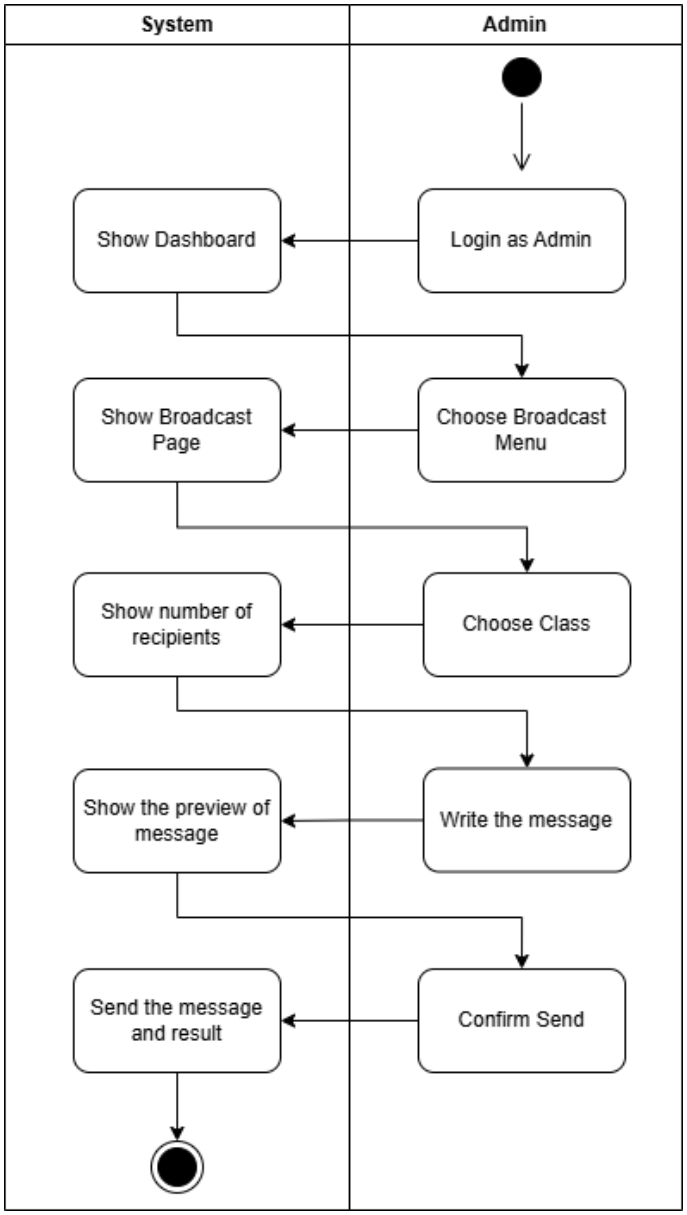

Figure 3.7 Activity Diagram for Admin Broadcasting WhatsApp

3.2.2.2 Teacher Activity Diagram

As illustrated in Figure 3.8, the activity diagram depicts the workflow for a teacher managing their own profile. The process begins when the teacher logs into the system, after which the system displays the teacher's profile information. The teacher may edit their personal data through the profile interface, and upon submission, the system validates and saves the updated information before displaying the revised profile to confirm that the changes have been successfully applied.

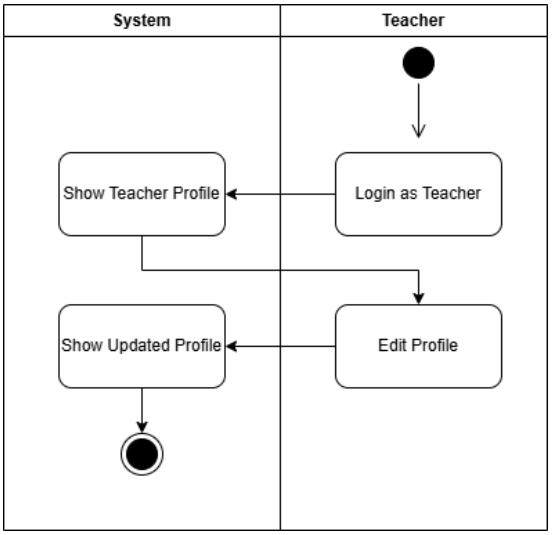

Figure 3.8 Activity Diagram for Teacher Profile Management

As illustrated in Figure 3.9, the activity diagram shows how a teacher manages the grades of their students. The teacher's profile and the classes and subjects they have been given are displayed by the system once they log in. Next, the instructor chooses the appropriate class and subject, then the academic year and semester. The matching student list is retrieved and shown by the system. The method automatically transforms a numerical score into a letter grade based on identified ranges: A (85–100), B (70–84), C (60–69), D (50–59), and E (0–49). The teacher can add or modify numeric ratings for each student. The teacher saves the grade after confirmation, and the system modifies the student grade records appropriately.

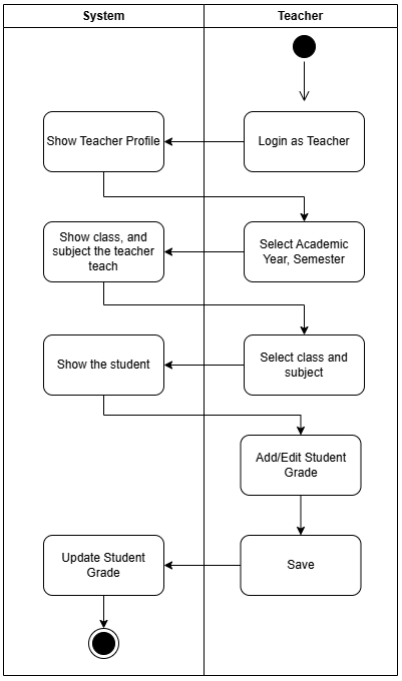

Figure 3.9 Activity Diagram for Teacher Managing Student Grades

As illustrated in Figure 3.10, the activity diagram depicts the workflow for teacher document management. The process begins when the teacher logs into the system, the teacher may choose to upload a new document or edit an existing one. In both cases, the system presents a document form in which the teacher selects the document type (e.g., lesson plan, teaching material) and uploads the corresponding file. After the form is submitted, the system validates the input, saves or updates the document data, and stores the uploaded file, completing the process.

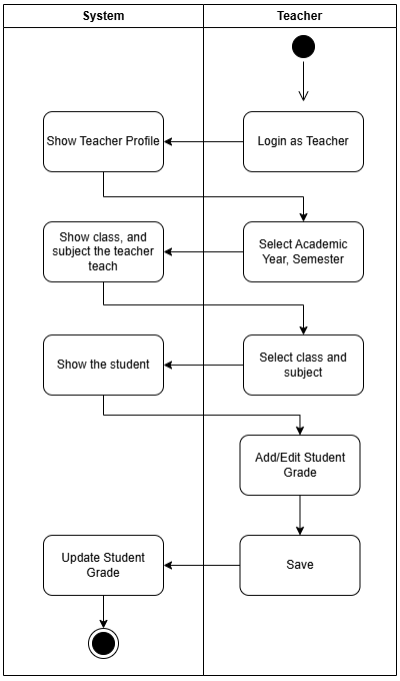

Figure 3.10 Activity Diagram for Teacher Document Management

3.2.2.3 Student Activity Diagram

As illustrated in Figure 3.11, the activity diagram shows the student profile management flow: the student logs in and views their profile, selects Edit Profile, and is presented with a form where Student ID, Email, Class, and Tahun Daftar are read-only. To change the password the student must enter the current password for verification. After submission the system validates editable fields, applies updates, and displays the revised profile.

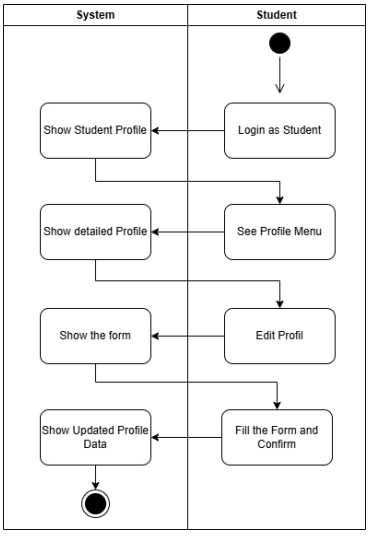

Figure 3.11 Activity Diagram for Student Profile Management

As illustrated in Figure 3.12, the student logs in and opens the Grades menu, then selects the desired class and academic year/semester. The system retrieves grade records filtered by the student's ID and the chosen class/term, converts numeric scores to letter grades where applicable, and displays the results. Empty or error states are handled with a clear message and options to refresh or export the report.

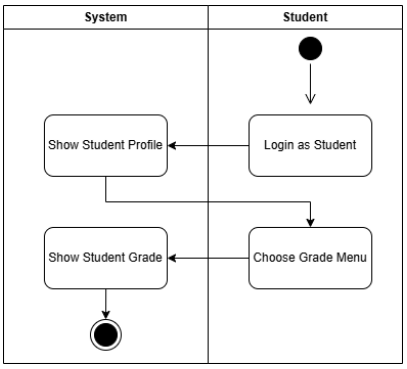

Figure 3.12 Activity Diagram for Student Viewing Grades

3.2.3 Database Design

The integrity of the data is maintained through application-level logic in Google Apps Script, which enforces relationships between entities using unique identifiers (Primary Keys). The database schema is organized into four primary entities to support academic operations: Student Data, Teacher Data, Academic Grades, and Class Schedules.

Given the serverless requirement of the system architecture, Google Sheets is utilized as the relational database backend. The database schema is organized into several logical tables to support core academic and administrative operations. The structure of these worksheets is detailed below:

a. Student Sheet (Student Data) stores all master data for students registered at the school. This table handles personal information, parent data, and student login credentials for system access. The Student Table structure can be seen in Table 3.1.

Table 3.1 Data Dictionary - Sheet "Student" (Data Siswa)

No	Attribute Name	Data Type	Key	Description
1	ID	String	PK	Unique Student Identifier (e.g., S-2025001).
2	NISN	String	-	National Student Registration Number.
3	Email	String	-	Student email address for login.
4	Password	String	-	Student login password.
5	Nama	String	-	Full name of the student.
6	Tempat_Lahir	String	-	City/Place of birth.
7	Tgl_Lahir	Date	-	Date of birth (DD/MM/YYYY).
8	Alamat	String	-	Residential address.
9	Gender	String	-	Gender (L/P).
10	Agama	String	-	Religion.
11	Kelas	String	FK	Foreign Key linking to Class ID (e.g., 1A).
12	Nama_Ayah	String	-	Father's full name.
13	Pekerjaan_Ayah	String	-	Father's occupation.
14	Nama_Ibu	String	-	Mother's full name.
15	Pekerjaan_Ibu	String	-	Mother's occupation.
16	No_HP_Ortu	String	-	Parent's WhatsApp number (e.g., 628...).
17	Tgl_Daftar	DateTime	-	Timestamp of registration.
18	Foto_URL	String	-	Google Drive link to student photo.

b. Teacher Sheet (Teacher Data), used to store teacher personnel data. This table includes information on Student ID Numbers (NIP), position, subjects taught, and employee status, which is used to validate logins in the teacher module. The Teacher table structure can be seen in Table 3.2.

Table 3.2 Data Dictionary - Sheet "Teacher"

No	Attribute Name	Data Type	Key	Description
1	ID	String	PK	Unique Teacher Identifier (e.g., G-001).
2	NIP	String	-	Employee Identification Number.
3	Email	String	-	Teacher email address for login.
4	Password	String	-	Teacher login password.
5	Nama	String	-	Full name of the teacher.
6	Gender	String	-	Gender (L/P).
7	Mapel	String	-	Primary subject taught (e.g., Matematika).
8	Status	String	-	Employment status (GTY/GTT).
9	Golongan	String	-	Civil servant rank (if applicable).
10	Pendidikan	String	-	Last education level (S1/S2).
11	Jabatan	String	-	Position (e.g., Wali Kelas, Guru Mapel).
12	No_HP	String	-	Teacher's WhatsApp number.
13	Foto_URL	String	-	Google Drive link to teacher profile photo.

c. NilaiSiswa Sheet (Grades Recap) serves as a transactional table that stores a history of student academic grades per semester. This table links student IDs to the subject grades entered by teachers. The NilaiSiswa table structure can be seen in Table 3.3.

Table 3.3 Data Dictionary - Sheet "NilaiSiswa"

No	Attribute Name	Data Type	Key	Description
1	ID_Siswa	String	FK	Link to Student ID (Column A).
2	Nama_Siswa	String	-	Name of student (for readability).
3	Semester	String	-	Academic Semester (1 or 2).
4	Kelas	String	-	Class name at the time of grading.
5	Nilai_Matematika	Integer	-	Numeric score for Math (0-100).
6	Huruf_Matematika	String	-	Letter grade for Math (A/B/C).
7	Nilai_IPA	Integer	-	Numeric score for Science (0-100).
8	Huruf_IPA	String	-	Letter grade for Science (A/B/C).
9	Nilai_B_Indo	Integer	-	Numeric score for Indonesian Language.
10	Huruf_B_Indo	String	-	Letter grade for Indonesian Language.
11	Nilai_Agama	Integer	-	Numeric score for Religious Studies.
12	Huruf_Agama	String	-	Letter grade for Religious Studies.
13	Tahun_Ajaran	String	-	Academic Year (e.g., 2025/2026).

d. JadwalSiswa Sheet (Lesson Schedule), designed to store the weekly lesson schedule configuration for each class. This table maps the days, lesson times, subjects, and teachers that will be displayed on the student dashboard. The JadwalSiswa table structure can be seen in Table 3.4.

Table 3.4 Data Dictionary - Sheet "JadwalSiswa"

No	Attribute Name	Data Type	Key	Description
1	ID_Jadwal	String	PK	Unique Schedule ID.
2	Kelas	String	FK	Target Class (e.g., 1A).
3	Hari	String	-	Day of the week (Senin-Jumat).
4	Jam_1_Mapel	String	-	Subject for 1st Period.
5	Jam_1_Guru	String	FK	Teacher Name/ID for 1st Period.
6	Jam_2_Mapel	String	-	Subject for 2nd Period.
7	Jam_2_Guru	String	FK	Teacher Name/ID for 2nd Period.
9	Jam_3_Mapel	String	-	Subject for 3rd Period.
10	Jam_3_Guru	String	FK	Teacher Name/ID for 3rd Period.
4	Jam_4_Mapel	String	-	Subject for 1st Period.
5	Jam_4_Guru	String	FK	Teacher Name/ID for 1st Period.
6	Jam_5_Mapel	String	-	Subject for 2nd Period.
7	Jam_5_Guru	String	FK	Teacher Name/ID for 2nd Period.
9	Jam_6_Mapel	String	-	Subject for 3rd Period.
10	Jam_6_Guru	String	FK	Teacher Name/ID for 3rd Period.
4	Jam_7_Mapel	String	-	Subject for 1st Period.
5	Jam_7_Guru	String	FK	Teacher Name/ID for 1st Period.
6	Jam_8_Mapel	String	-	Subject for 2nd Period.
7	Jam_8_Guru	String	FK	Teacher Name/ID for 2nd Period.

3.2.4 Entity Relationship Diagram

The logical data model for the system is formalized in the Entity Relationship Diagram (ERD) presented in Figure 3.13. This diagram translates the Google Sheets schema from Table 3.1 into a standard relational model, illustrating the core entities—Student, Teacher, Kelas, nilaiSiswa, jadwalSiswa, and teacherDocument—and the fundamental relationships between them.

The ERD employs Crow's Foot notation to define the cardinality of each relationship, which are central to the application's logic:

- Student and nilaiSiswa (Grades): A Student can have multiple Nilai_Siswa records across different semesters and academic years (1:N relationship). Each grade record is uniquely tied to one student.
- Teacher and teacherDocument: A Teacher can upload multiple Teacher_Document records, such as lesson plans (RPP) (1:N relationship). Each document is associated with a single teacher.
- Teacher and jadwalSiswa (Schedule): A Teacher can be assigned to teach in multiple schedule slots within the Jadwal_Siswa entity (1:N relationship). This relationship is implemented through the GURU_ID and GURU_nama foreign key attributes in the schedule.

This visual model serves as the authoritative blueprint for the system's data architecture. It confirms that the spreadsheet-based implementation, while utilizing Google Sheets as the storage layer, successfully enforces the necessary relational integrity to support SD Muhammadiyah Kadisoka's operational workflows, from grade management and document tracking to schedule administration.

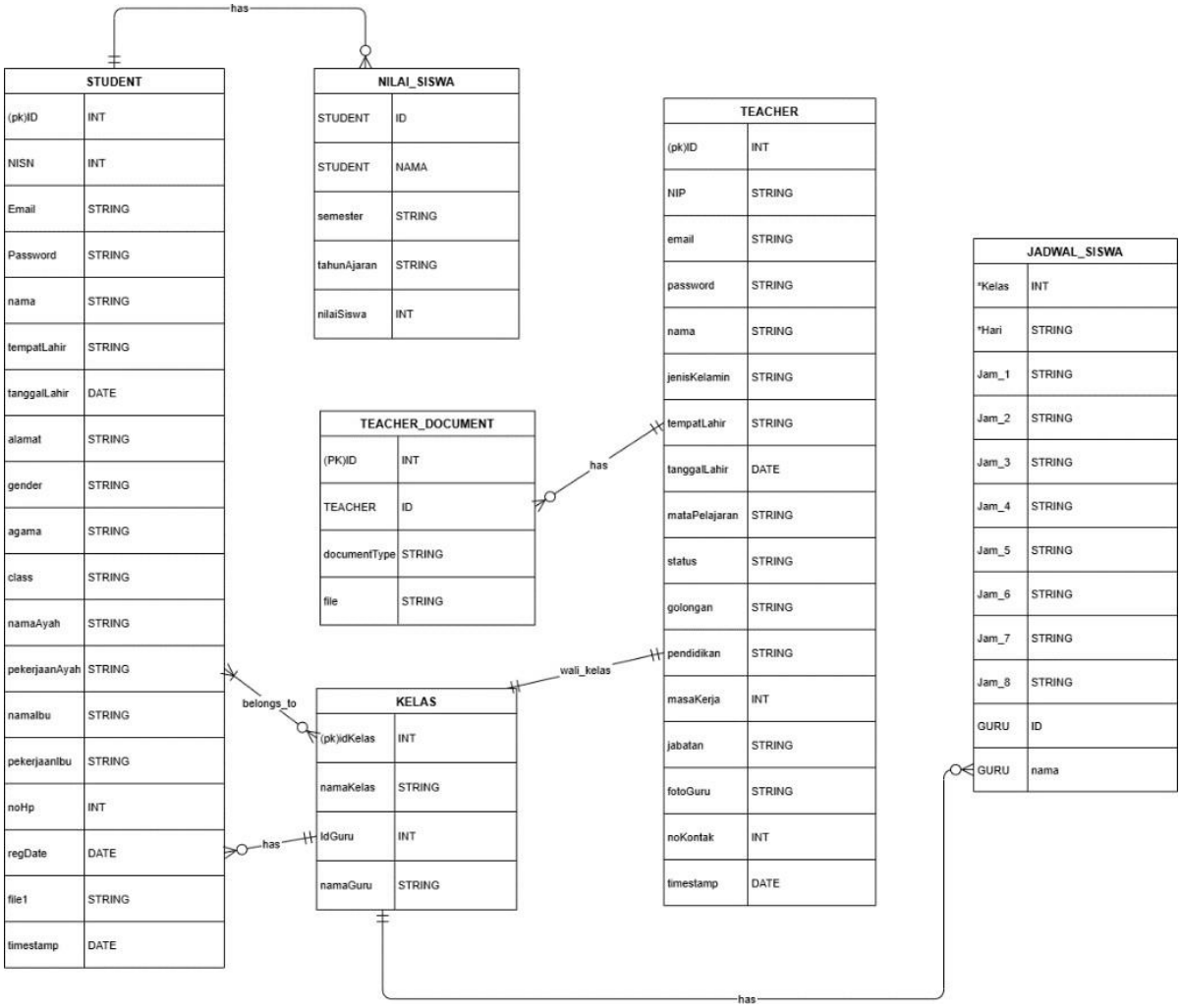

Figure 3.13 Entity Relationship Diagram of the School Management Information System

3.3 System Implementation Environment

The system is developed using a serverless stack within the Google Workspace ecosystem, ensuring zero cost for hosting and maintenance.

a. Backend: Google Apps Script (GAS) serves as the controller, handling HTTP requests (doGet, doPost) and business logic.

b. Database: Google Sheets acts as the data persistence layer.

c. Storage: Google Drive is used for storing binary files (Student Photos and Teacher Lesson Plans).
d. Frontend: HTML5, CSS3, and JavaScript are served directly from GAS to the client browser.
e. Notification Gateway: The Wablas API is integrated to facilitate automated WhatsApp broadcasts. The logic is programmed to handle API throttling, ensuring messages are sent with a safety delay (20–30 seconds) to prevent blocking.

3.4 Testing and Evaluation Methodology

To validate the system, two distinct testing methodologies are employed: Functional Testing (Black-Box) and Usability Testing (SUS).

3.4.1 Black-Box Testing

Black-Box testing focuses on the functional specifications of the software. Test cases are designed to verify that inputs produce the expected outputs without examining the internal code structure (Ostrand, 2002). The testing scenarios include:

a. Admin Module: verification of Add/Edit/Delete functions for Student, Teacher, Class, and Schedule data.
b. Teacher Module: Verification of Grade inputs, Profile edits, and Document (RPP) uploads.
c. Student Module: Verification of Dashboard loading, Schedule retrieval, and Grade viewing.
d. Performance: Measuring system response times.

3.4.2 System Usability Scale (SUS)

To measure user satisfaction and the system's ease of use, the System Usability Scale (SUS) method is applied involving the School Principal, Teachers, Administrative Staff, and Student Guardians. Respondents are asked to rate the 10 standard statements on a 5-point Likert scale ranging from "Strongly Disagree" (1) to "Strongly Agree" (5). The specific SUS items used in this evaluation are listed in Table 3.2.

Table 3.5 Summary of SUS Scores by User Group

ID	SUS Question
Q1	I think I will use this system regularly to manage or check school needs.
Q2	This system feels rigid and less practical when used for school purposes.
Q3	I found this system easy to use.
Q4	I felt I needed help from another person, staff, or IT specialist to use it.
Q5	The menus (grades, schedules, data) are well-organized and interconnected.
Q6	The system's appearance and functionality are variable and inconsistent.
Q7	I'm confident most people can learn to use this system quickly.
Q8	I think the system is too complicated, even though it could be simpler.
Q9	I feel very confident using this system.
Q10	I need to learn a lot before I can start using this system smoothly.

Scoring calculation can be seen in equation (3.1), (3.2), (3.3).The final SUS score is calculated using the standard formula.

- For odd-numbered items (1, 3, 5, 7, 9):

$$\text{Score} = \text{User Response} - 1 \quad (3.1)$$

- For even-numbered items (2, 4, 6, 8, 10):

$$\text{Score} = 5 - \text{User Response} \quad (3.2)$$

$$\text{Final Score} = \text{User Response} - 1 \quad (3.3)$$

This calculation results in a score between 0 and 100, which is then mapped to usability adjective ratings (e.g., "Good", "Excellent") to determine the acceptability of the system at SD Muhammadiyah Kadisoka.

CHAPTER IV
SYSTEM IMPLEMENTATION AND TESTING

This chapter details the technical realization and evaluation of the cloud-based School Management Information System (MIS) developed for SD Muhammadiyah Kadisoka. Following the design and requirements analysis phases outlined in Chapter III, the system was built to address the specific operational constraints identified during the preliminary research. The implementation focuses on the primary objectives derived from stakeholder interviews: providing a digital platform for teacher document submissions, eliminating reliance on expensive local servers, and extending data accessibility to parents who previously had no direct access beyond the government-centralized Dapodik platform. This chapter presents the database construction, interface development, system integration, and the results of both functional (Black-Box) and usability (System Usability Scale) testing.

4.1 Database and System Implementation

The system backend was constructed entirely within the Google Workspace ecosystem, adhering to the serverless, zero-cost architecture defined in the methodology. Google Sheets was utilized as the primary structured data store, with separate worksheets acting as logical database tables for each core entity.

To concretely demonstrate how the system design described in Chapter III was realized, the implementation details are presented through a combination of code excerpts and interface screenshots. These figures are not intended to document the full source code, but to highlight representative logic and mechanisms that implement authentication, data processing, external integration, and role-based access control within a serverless environment.

4.1.1 Database Implementation

The database was implemented directly within Google Sheets, as outlined in the design phase. The structure of the key worksheets is as follows:

a. Student Sheet: This sheet stores records for all students. Each row contains the student's unique ID, NISN, email, password, full name, birth details, address, gender, religion, class, parent information (names, occupations, contact number), registration date, a URL link to the student's photo stored in Google Drive, and a timestamp.

b. Teacher Sheet: This sheet manages data teachers. Columns include teacher ID, NIP, email, password, personal details, assigned subject (Mata Pelajaran), employment status, rank (Golongan), education level, years of service, position (Jabatan), photo URL, contact number, and timestamp.
c. listKelas Sheet: This sheet defines the 18 classes (e.g., 1A, 1B, 1C, 2A...). It links each class (Nama Kelas) to a homeroom teacher (Wali Kelas) via the teacher's ID and name.
d. nilaiSiswa Sheet: This sheet is structured for grade management. For each student and academic period, it records numeric and letter grades for each subject (e.g., Mathematics, Science, Religion). The structure includes columns for Student ID, Name, Semester, Class, followed by subject-grade pairs (e.g., Matematika (minat), Nilai Huruf), and the Tahun Ajaran.
e. jadwalSiswa Sheet: This sheet stores the weekly schedule. Each row defines the timetable for a specific class and day, detailing eight periods (jam pertama to jam ke 8), each with its subject and assigned teacher.
f. teacherDocument Sheet: This sheet records documents submitted by teachers, including the document type, teacher ID and name, a Google Drive link to the file, and a timestamp.

This schema, implemented in a familiar spreadsheet interface, allows non-technical administrative staff at SD Muhammadiyah Kadisoka to view and perform basic data management directly in Google Sheets, providing a fallback mechanism and enhancing transparency.

4.1.2 Implementation of Backend Logic

The backend of the School Management Information System (MIS) for SD Muhammadiyah Kadisoka is built entirely on Google Apps Script. This serverless environment acts as the middleware, managing HTTP requests from the client side, executing business logic, and interacting with the Google Sheets database and the Wablas API. To ensure code maintainability and modularity, the implementation is divided into five core logical segments: Request Routing and Authentication, Master Data Management, WhatsApp Notification Gateway, Academic Grading Logic, and Student Data Retrieval.

4.1.2.1 Request Routing and Session Authentication

The authentication and request-routing mechanism forms the foundation of secure system access. As illustrated in Figure 4.1, incoming HTTP requests are handled by the Google

Apps Script entry point and routed dynamically based on URL parameters. User credentials are validated against role-specific data sources to ensure that administrators, teachers, and students are granted access only to their respective system interfaces.

```
// 1. HTTP REQUEST HANDLING
function doGet(e) {
  let page = e.parameter.page;
  if (page == null) page = "index";
  var output = HtmlService.createTemplateFromFile(page);
  return output.evaluate()
    .addMetaTag('viewport', 'width=device-width , initial-scale=1')
    .setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL);
}

function include(filename) {
  return
HtmlService.createTemplateFromFile(filename).evaluate().getContent();
}

// 2. SESSION AND LOGIN LOGIC
function checkLogin(email, password) {
  // Hard-coded admin credentials for Root Security
  var validAdminEmail = '***@email.com';
  var validAdminPassword = '***';

  // A. Check Admin
  if (email.toLowerCase() === validAdminEmail.toLowerCase() &&
      password === validAdminPassword) {
    var userProperties = PropertiesService.getUserProperties();
    userProperties.setProperty('userEmail', email);
    userProperties.setProperty('userRole', 'admin');
    return 'ADMIN';
  }

  // B. Check Teacher (Iterative Lookup)
  var teacherData = getAllTeacherData();
  for (var i = 0; i < teacherData.length; i++) {
    if (teacherData[i][2] === email && teacherData[i][3] === password) {
      var userProperties = PropertiesService.getUserProperties();
      userProperties.setProperty('userEmail', email);
      userProperties.setProperty('userRole', 'teacher');
      return 'TEACHER';
    }
  }

  // C. Check Student
  var studentData = getAllData();
  for (var j = 0; j < studentData.length; j++) {
    if (studentData[j][2] === email && studentData[j][3] === password) {
      var userProperties = PropertiesService.getUserProperties();
      userProperties.setProperty('userEmail', email);
      userProperties.setProperty('userRole', 'student');
      return 'STUDENT';
    }
  }
  return 'FALSE';
}
```

```
function logoutUser() {
  try {
    PropertiesService.getUserProperties().deleteAllProperties();
    return true;
  } catch (error) {
    return false;
  }
}
```

Figure 4.1 Request Routing and Authentication

4.1.2.2 Master Data Management

Following successful authentication, core administrative operations are executed through modular backend functions. The system utilizes a custom set of helper functions to perform Create, Read, Update, and Delete (CRUD) operations on the Google Sheets database. The implementation of student master data management is shown in Figure 4.2, which demonstrates the logic for generating unique identifiers, validating input data, storing records in Google Sheets, and handling file uploads to Google Drive. This module ensures data consistency while maintaining usability for non-technical administrators.

In terms of system limitations, the CRUD module is also the area most affected by Google Apps Script execution quotas and runtime limits. Administrative operations such as loading full datasets, performing row-by-row validation, generating IDs by scanning historical records, and updating multiple spreadsheet ranges may become computationally expensive as the dataset grows. Since Google Apps Script enforces a maximum runtime per execution, bulk CRUD operations that involve large-range scanning or repeated write calls risk being interrupted if they exceed execution constraints. However, within the operational scale of SD Muhammadiyah Kadisoka (approximately 500 students and 56 teachers), these limitations did not prevent successful data processing during testing. This confirms that the architecture remains practical for low-to-moderate school-scale transactions, while still requiring careful design if the system expands significantly in the future.

```
  // CONFIGURATION
const SPREADSHEET_ID = '***';

// 3. STUDENT DATA MANAGEMENT
function globalVariables() {
  return {
    spreadsheetId: SPREADSHEET_ID,
    dataRange: 'Student!A3:S',
    idRange: 'Student!A2:S',
    lastCol: 'S',
```

```
    insertRange: 'Student!A1:S1'
  };
}

function generateID() {
  const                                sheet                                =
SpreadsheetApp.openById(SPREADSHEET_ID).getSheetByName("Student");
  const          idData          =          sheet.getRange("A3:A"          +
sheet.getLastRow()).getValues().flat().filter(String);
  let maxNumber = 0;

  idData.forEach(id => {
    const number = parseInt(id.replace("S-", ""), 10);
    if (!isNaN(number) && number > maxNumber) maxNumber = number;
  });
  return `S-${maxNumber + 1}`;
}

function processForm(formObject) {
  const vars = globalVariables();
  let isNew = false;

  // Handle File Upload to Google Drive
  let fileUrl = "";
  if (formObject.myFile1 && formObject.myFile1.length > 0) {
      var folder = DriveApp.getFolderById('***');
      var blob = formObject.myFile1;
      var file = folder.createFile(blob);
      fileUrl = file.getUrl();
  }

  // Prepare Data Array
  const values = [[
      (formObject.RecId && checkID(formObject.RecId)) ? formObject.RecId :
generateID(),
      formObject.nisn,         formObject.email,         formObject.password,
formObject.nama,
      formObject.tempatlahir, formObject.tglLahir, formObject.alamatAsal,
      formObject.gender, formObject.agama, formObject.kelas,
      formObject.fatherName, formObject.pekerjaanAyah, formObject.namaIbu,
      formObject.pekerjaanIbu, formObject.noHp, formObject.regYear,
      fileUrl, new Date().toLocaleString()
  ]];

  // Logic: Update if ID exists, Append if New
  if (formObject.RecId && checkID(formObject.RecId)) {
    updateData(values, vars.spreadsheetId, getRangeByID(formObject.RecId));
  } else {
    appendData(values, vars.spreadsheetId, vars.insertRange);
    isNew = true;
  }

  // Trigger Welcome Message if New
  if (isNew) {
     sendWelcomeWA(formObject.noHp,    formObject.nama,    formObject.email,
formObject.password);
  }
  return getAllData();
}
```

```
// HELPER CRUD FUNCTIONS
function appendData(values, spreadsheetId, range) {
  var valueRange = Sheets.newRowData();
  valueRange.values = values;
  var request = Sheets.newAppendCellsRequest();
  request.sheetID = spreadsheetId;
  request.rows = valueRange;
  Sheets.Spreadsheets.Values.append(valueRange,     spreadsheetId,     range,
{valueInputOption: "RAW"});
}

function updateData(values, spreadsheetId, range) {
  var valueRange = Sheets.newValueRange();
  valueRange.values = values;
  Sheets.Spreadsheets.Values.update(valueRange,     spreadsheetId,     range,
{valueInputOption: "RAW"});
}
```

Figure 4.2 Master Data Processing and ID Generation

4.1.2.3 WhatsApp Notification Gateway

To facilitate real-time communication between the school and parents, the backend integrates with the Wablas API. A critical technical challenge in this gateway integration is compliance with Wablas/WhatsApp rate limiting and anti-spam safeguards. If messages are dispatched too rapidly, the API may throttle requests or WhatsApp may flag the delivery behavior as repetitive broadcasting.

To address this constraint, the system implements a throttling mechanism using Utilities.sleep(1500), adding a 1.5-second delay per recipient. This approach ensures controlled delivery, reduces the likelihood of request rejection, and maintains stable broadcast success rates under normal school-scale usage. Although throttling causes messages to be sent sequentially rather than instantly, this trade-off improves reliability for announcements delivered to selected classes or the entire school population.Figure 4.3 presents the broadcast logic that supports this controlled message delivery strategy.

```
  // 4. WHATSAPP GATEWAY INTEGRATION
const urlWA = 'https://jogja.wablas.com/api/v2/send-message';
const tokenWA = '***.***';

function getStudentsByClass(className) {
  const data = getAllData(); // Fetch all students
  if (className === 'all') return data;
  // Filter by Class (Column Q / Index 16)
  return  data.filter(row  =>  row[16]  &&  row[16].toString().trim()  ===
className);
}

function sendBroadcast(message, className) {
```

```
  try {
    const students = getStudentsByClass(className);
    const results = { success: 0, failed: 0 };

    for (let i = 0; i < students.length; i++) {
      let student = students[i];
      if (!student) continue;

      const phone = student[15]; // Column P
      const name = student[4];   // Column E

      if (phone) {
        // Format Phone Number (08xxx -> 628xxx)
        let noHp = phone.toString().replace(/\D/g, '');
        if (noHp.startsWith("0")) noHp = "62" + noHp.slice(1);

        const payload = JSON.stringify({
          data: [{ phone: noHp, message: message }]
        });

        const options = {
          method: 'POST',
          headers:     {     Authorization:     tokenWA,     'Content-Type':
'application/json' },
          payload: payload,
          muteHttpExceptions: true
        };

        const response = UrlFetchApp.fetch(urlWA, options);
        const respData = JSON.parse(response.getContentText());

        if (respData.status) results.success++; else results.failed++;
      }

      // CRITICAL: Throttling to prevent blocking (1.5s delay per message)
      if (i < students.length - 1) Utilities.sleep(1500);
    }
    return results;
  } catch (error) {
    return { error: error.toString() };
  }
}
```

Figure 4.3 WhatsApp Broadcast Implementation with Rate Limiting

4.1.2.4 Teacher Grading and Duplicate Prevention

The grading module is the most complex logical component of the system. It must handle two scenarios: inserting a new grade for a student or updating an existing one. As shown in Figure 4.4, the updateMultipleGrades function iterates through the NilaiSiswa sheet to check if a record already exists for the specific combination of Student ID, Semester, and Academic Year. This logic prevents data redundancy (duplicate rows) in the spreadsheet database.

```
  //TEACHER GRADING MODULE
```

```
function updateMultipleGrades(gradeUpdates, teacherEmail, semester,
tahunAjaran) {
  const ss = SpreadsheetApp.openById(SPREADSHEET_ID);
  const gradeSheet = ss.getSheetByName("nilaiSiswa");
  const allData = gradeSheet.getDataRange().getValues();
  const existingGrades = allData.slice(2); // Skip headers

  // Mapping Subject Names to Column Indexes
  const subjectColumnMap = {
    "Matematika (minat)": 4, "Fisika": 6, "Kimia": 8, "Biologi": 10,
    "Ekonomi": 12, "Geografi": 14, "Bahasa Indonesia": 20, "Matematika
(umum)": 28,
    "Pendidikan Agama": 24, "PPKn": 26, "Informatika / Prakarya": 34
    // ... other subjects mapped similarly
  };

  let newRows = [];

  gradeUpdates.forEach(update => {
    const subjectCol = subjectColumnMap[update.subject];
    const letterGradeCol = subjectCol + 1;

    if (subjectCol) {
      // Check if Grade Record Exists
      let existingRowIndex = -1;
      for (let i = 0; i < existingGrades.length; i++) {
        if (existingGrades[i][0] === update.studentID &&
            existingGrades[i][2] == semester &&
            existingGrades[i][36] === tahunAjaran) {
          existingRowIndex = i;
          break;
        }
      }

      if (existingRowIndex !== -1) {
        // CASE 1: UPDATE EXISTING ROW
        const actualRow = existingRowIndex + 3;
        gradeSheet.getRange(actualRow,              subjectCol              +
1).setValue(update.numericGrade);
        gradeSheet.getRange(actualRow,            letterGradeCol            +
1).setValue(update.letterGrade);
      } else {
        // CASE 2: PREPARE NEW ROW (If not already in batch)
        const  existingNewRowIndex  =  newRows.findIndex(r  =>  r[0]  ===
update.studentID);

        if (existingNewRowIndex !== -1) {
           newRows[existingNewRowIndex][subjectCol] = update.numericGrade;
        } else {
           // Get Student Name from Master Data
           const   studentData   =   getAllData().find(r   =>   r[0]   ===
update.studentID);
           if (studentData) {
             const newRow = new Array(37).fill('');
             newRow[0] = update.studentID;
             newRow[1] = studentData[4]; // Name
             newRow[2] = semester;
             newRow[3] = update.kelas;
             newRow[subjectCol] = update.numericGrade;
             newRow[letterGradeCol] = update.letterGrade;
```

```
              newRow[36] = tahunAjaran;
              newRows.push(newRow);
            }
          }
        }
      }
    });

    // Batch Insert New Rows
    if (newRows.length > 0) {
      const nextRow = existingGrades.length + 3;
      gradeSheet.getRange(nextRow, 1, newRows.length, 37).setValues(newRows);
    }
}
```

Figure 4.4 Grading Logic with Duplicate Entry Prevention

4.1.2.5 Schedule Data Retrieval for Student

The system ensures that student guardians can access specific data without seeing the entire school database. Figure 4.5 demonstrates the getStudentGrades and getStudentSchedule functions. These functions filter the master datasets based on the logged-in user's credentials (email/ID) and return only the relevant JSON objects to the frontend, ensuring data privacy and efficient loading.

```
  // 6. STUDENT PORTAL DATA FETCHING
function getStudentGrades(studentId, academicYear, semester, studentClass)
{
  const ss = SpreadsheetApp.openById(SPREADSHEET_ID);
  const sheet = ss.getSheetByName('nilaiSiswa');
  const data = sheet.getDataRange().getDisplayValues();
  const headers = data[0];

  for (let i = 1; i < data.length; i++) {
    // Match Student ID, Year, and Semester
    if (data[i][0] == studentId &&
        (data[i][30] || '2025/2026') == academicYear &&
        (data[i][2] || '1') == semester) {

        const grades = {};
        for (let j = 4; j < headers.length; j += 2) {
           const subject = headers[j];
           if (subject && subject !== 'Tahun Ajaran') {
             grades[subject] = {
               numeric: data[i][j] || 'n/a',
               letter: data[i][j+1] || 'n/a'
             };
           }
        }
        return { grades: grades };
    }
  }
  return { error: 'No grades found for this period.' };
}
```

```
function getStudentSchedule(studentEmail) {
  // 1. Get Class from Student Profile
  const studentData = getAllData().find(row => row[2] === studentEmail);
  if (!studentData) return [];
  const studentClass = studentData[10];

  // 2. Fetch Schedule for that Class
  const                                sheet                                =
SpreadsheetApp.openById(SPREADSHEET_ID).getSheetByName("jadwalSiswa");
  const data = sheet.getDataRange().getDisplayValues();
  const headers = data[0];
  const schedule = [];

  for (let i = 1; i < data.length; i++) {
    if (data[i][1] === studentClass) {
       const item = {};
       headers.forEach((h, idx) => item[h] = data[i][idx]);
       schedule.push(item);
    }
  }
  return schedule;
}
```

Figure 4.5 Data Retrieval Logic

4.1.3 Implementation of Backend Logic

The core functionality of the system is built upon Google Apps Script, which functions as a serverless middleware connecting the client-side user interface (HTML/JavaScript) with Google Sheets and Google Drive. The backend logic is divided into four primary modules: Request Routing & Authentication, CRUD, External API Integration, and File Management.

4.1.4 User Interface Implementation

The User Interface (UI) was built using HTML, CSS, and JavaScript, served dynamically by Google Apps Script. The frontend adapts its content and functionality based on the authenticated user's role, providing three distinct operational views.

4.1.4.1 Admin Module

Admin Module functions as the system’s central control hub. Upon accessing the system, admin first authenticates through the login page shown in Figure 4.6. After successful login, admin is presented with a dashboard that visualizes key school metrics—such as total student enrollment, teacher count, and student distribution by gender and class—as illustrated in Figure 4.7. This module provides Create, Read, Update, and Delete (CRUD) capabilities for all master data. The Student and Teacher Data Management interface illustrated in Figure 4.8 enables

admin to register new students and teachers, update existing records, or deactivate entries. During the registration of new students or teachers. Admin can also disseminate announcements using the broadcast feature shown in Figure 4.9, which supports delivery to selected classes or all users. Class schedules are configured through the Schedule Management interface presented in Figure 4.10, where subjects and teachers are assigned to class time slots.

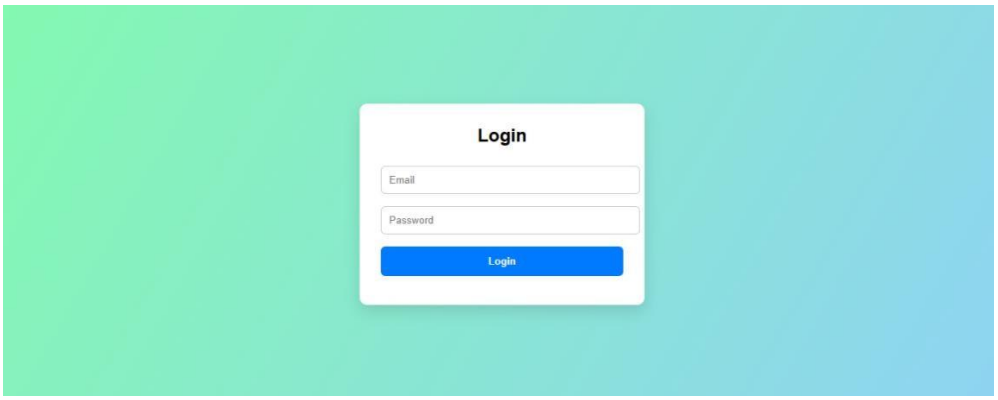

Figure 4.6 Administrator Login Page

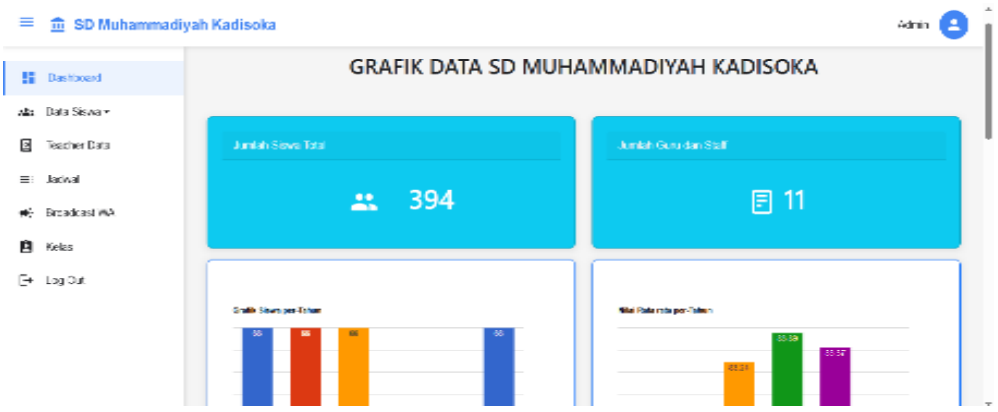

Figure 4.7 Administrator Dashboard Interface

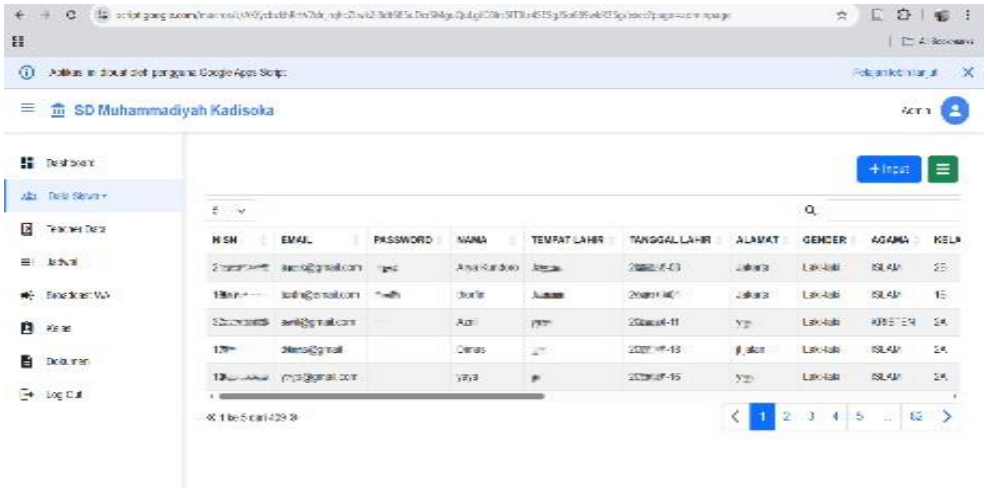

Figure 4.8 Student Data Management Interface

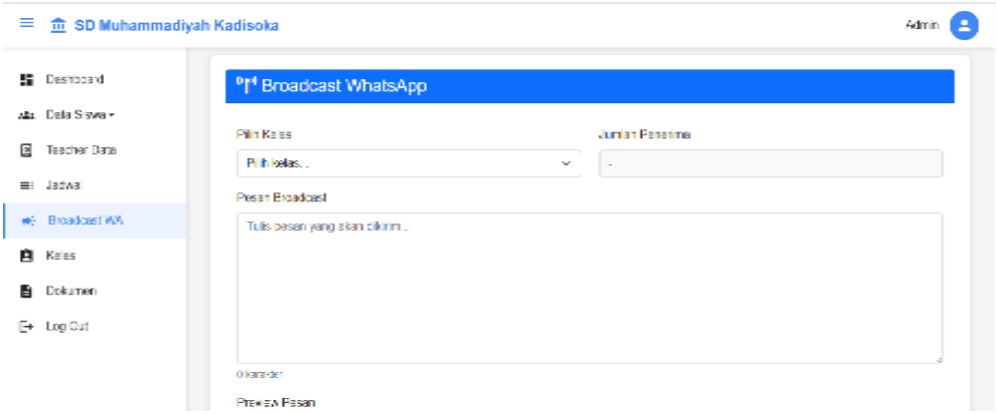

Figure 4.9 Administrator Broadcast Interface

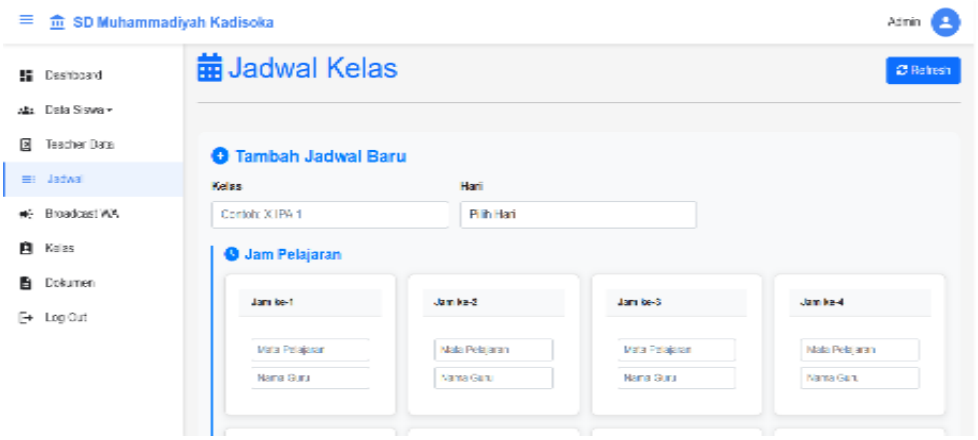

Figure 4.10 Schedule Management Interface

4.1.4.2 Teacher Profile

This module addresses the gap identified during interviews, the absence of a digital platform to support daily teaching activities. Teachers begin by authenticating through the login interface shown in Figure 4.11. After logging in, teachers can view and update their personal profiles, illustrated in Figure 4.12. The grade feature presented in Figure 4.13, which dynamically loads the list of students based on the classes and subjects assigned to the teacher. Teachers enter numerical grades, and the system automatically converts them into letter grades according to predefined ranges. In addition, teachers can upload lesson plans (RPP) and other teaching documents, which are securely stored in a designated Google Drive folder, replacing the conventional physical submission process.

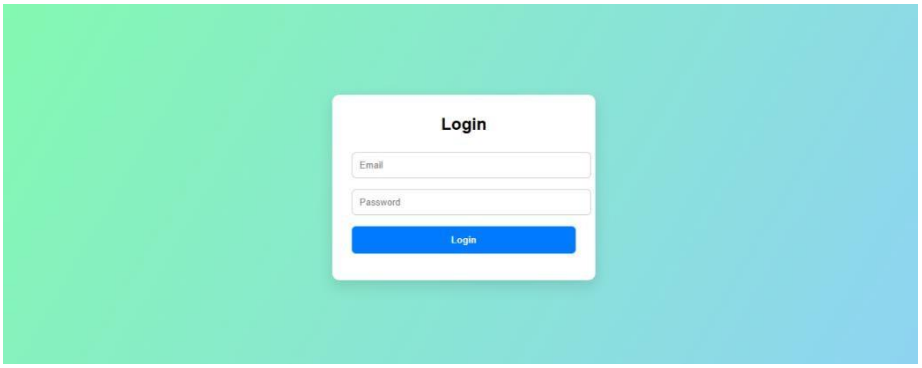

Figure 4.11 Teacher Login Interface

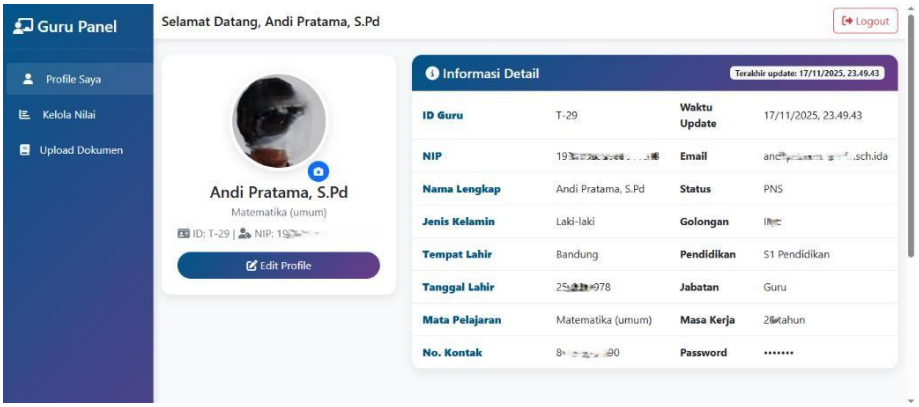

Figure 4.12 Teacher Profile Management Interface

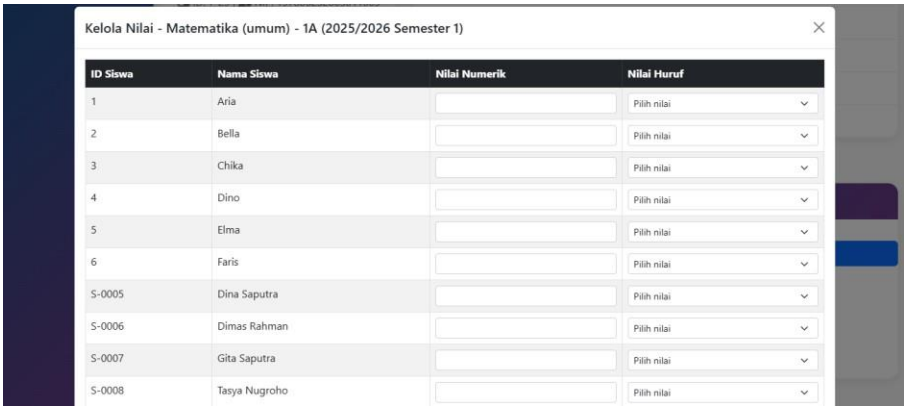

Figure 4.13 Teacher Grade Input Interface

4.1.4.3 Student Profile

This module provides the transparency that was previously lacking for students. Students or guardians authenticate through the login interface shown in Figure 4.14. After logging in, users can view their profile information, as illustrated in Figure 4.15, and access two key academic features: the student's detailed class schedule and grade reports. The academic schedule displayed in Figure 4.16 is retrieved in real time from the system based on the student's assigned class. The grade report interface shown in Figure 4.17 allows guardians to select the relevant academic year and semester to view the corresponding grades. This module fulfills the objective of providing direct, self-service access to academic information without requiring mediation from school staff.

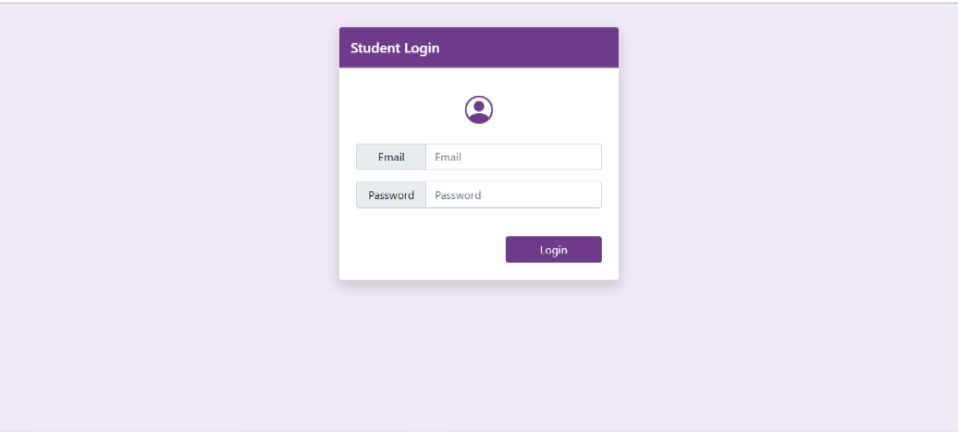

Figure 4.14 Student Login Interface

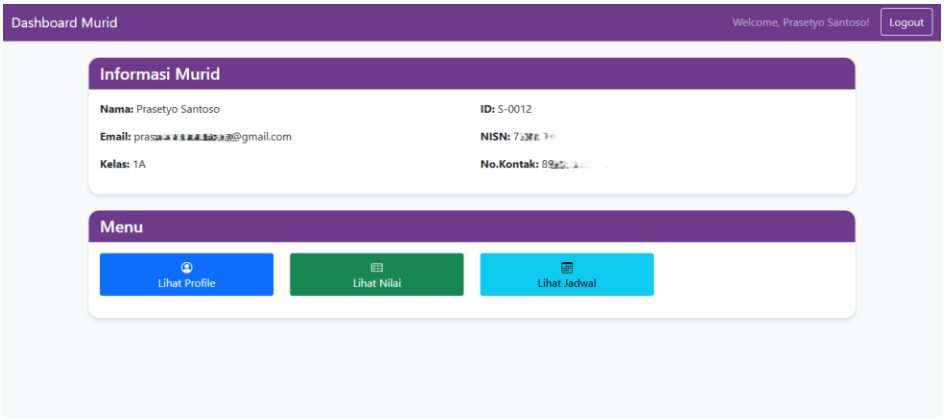

Figure 4.15 Student Profile View Interface

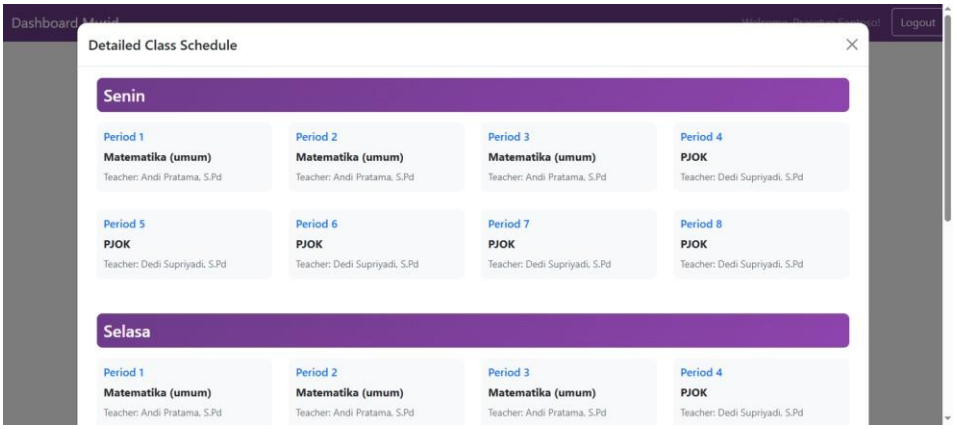

Figure 4.16 Student Academic Schedule Interface

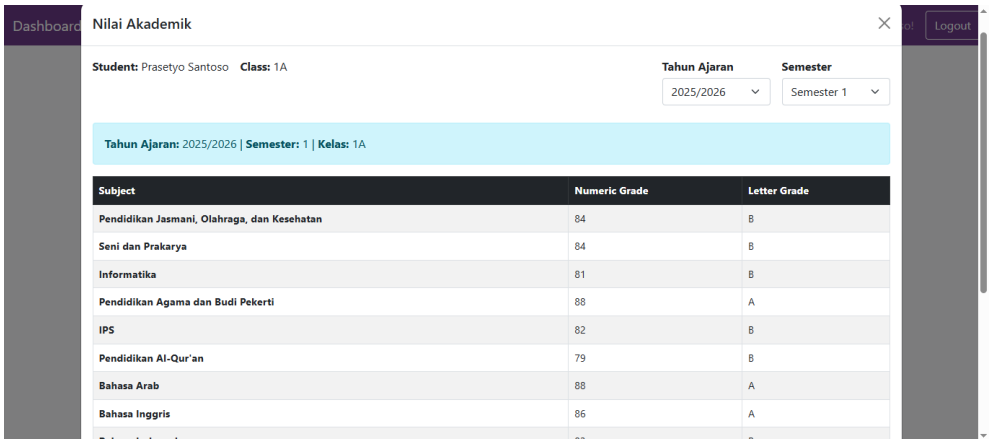

Figure 4.17 Student Grade Report Interface

4.2 Notification System Integration

To address communication needs and leverage widespread WhatsApp usage in Indonesia, the system integrates the Wablas API as an automated notification gateway within

the Google Apps Script backend. Notifications are triggered by specific events; for instance, when an administrator registers a new student, the system generates a JSON payload containing the formatted parent phone number and a personalized welcome message with login credentials, as shown in Figure 4.18, and sends it via an HTTP POST request to the Wablas API. Administrators can also send broadcast messages to all parents or selected classes through the admin interface. Messages are dispatched sequentially with a 20–30 second delay to comply with API fair-use policies, and delivery confirmations are logged and displayed to the administrator.

Figure 4.18 WhatsApp Welcome Message Delivery Result

4.3 Testing Process

Testing was conducted to validate that the system meets the operational requirements of SD Muhammadiyah Kadisoka. The testing phase is divided into Functional Testing (Black-Box), Performance Testing, and Usability Testing (SUS).

4.3.1 Functional Testing (Black-Box)

In this phase, functional testing was performed to verify that the application operates according to the specifications defined in Chapter III. The testing method employed is Black-Box Testing, which focuses on the inputs and expected outputs of the software without inspecting the internal code structure. This ensures that every function from authentication to complex grading logic delivers the correct result to the end-user.

The testing scenarios are categorized based on the user roles identified in the system analysis: Administrator, Teacher, and Student/Guardian. The following subsections present the detailed results of these test cases.

4.3.1.1 Authentication and Security Testing

The first stage of testing focused on the security of the system, specifically the login mechanism and role-based routing. This ensures that users (Admin, Teachers, and Students) are correctly identified and redirected to their respective dashboards. The results of the authentication testing are presented in Table 4.1.

Table 4.1 Testing of Login Authentication

Test ID	Scenario	Test Input Data	Expected Result	Actual Result	Status
TC-001	Login with empty fields	Email: [Empty] Pass: [Empty]	System displays alert "Harap isi email dan password".	Alert displayed.	Valid
TC-002	Login with unregistered email	Email: unknown@mail.com Pass: 12345	System displays alert "User tidak ditemukan".	Alert displayed.	Valid
TC-003	Login with wrong password	Email: admin@sdislam.id Pass: wrongpass	System displays alert "Password salah".	Alert displayed.	Valid
TC-004	Admin Login (Success)	Email: admin@sdislam.id Pass: admin123	System redirects to Admin Dashboard.	Redirected to Admin Dashboard.	Valid
TC-005	Teacher Login (Success)	Email: guru01@sdislam.id Pass: guru123	System redirects to Teacher Profile.	Redirected to Teacher Profile.	Valid
TC-006	Student Login (Success)	Email: siswa01@sdislam.id Pass: siswa123	System redirects to Student Profile.	Redirected to Student Profile.	Valid

4.3.1.2 Admin Module Testing

The second stage validated the Administrator's capability to manage master data, including creating, reading, updating, and deleting (CRUD) records for Students, Teachers, and Classes. This section also tested the system's ability to handle file uploads (photos) and

integrate with Google Drive. The results of the Admin module testing are summarized in Table 4.2.

Table 4.2 Testing of Administrator Data Management

Test ID	Scenario	Test Input Data	Expected Result	Actual Result	Status
TC-008	Add New Student	Complete form data (Name, NISN, Class, etc.)	Data saved to "Student" Sheet; ID generated automatically.	Data saved; ID "S-001" created.	**Valid**
TC-009	Upload Student Photo	Image file (JPG, 1MB)	File saved to Drive; URL saved to Sheet.	Link appears in column "Foto_URL".	**Valid**
TC-010	Edit Student Data	Change "Alamat" for Student S-001	Data in Sheet updates without creating new row.	Address updated correctly.	**Valid**
TC-011	Delete Student	Select Student S-001 -> Click Delete	Row containing S-001 is removed from Sheet.	Row deleted.	**Valid**
TC-012	Add New Teacher	Complete form data (Name, NIP, Mapel)	Data saved to "Teacher" Sheet.	Data saved successfully.	**Valid**
TC-013	Manage Class Data	Assign "Guru A" as Wali Kelas for "1A"	"ListKelas" Sheet updates with new Wali Kelas.	Sheet updated.	**Valid**

4.3.1.3 Teacher Module Testing

The third stage focused on the Teacher's workflow, specifically the grading system and document management. Tests were conducted to verify that numeric grades are automatically and accurately converted into letter grades (A, B, C, etc.) and that Lesson Plans (RPP) are successfully uploaded to the system. The functional test results for the Teacher module are shown in Table 4.3.

Table 4.3 Testing of Teacher Profile Features

Test ID	Scenario	Test Input Data	Expected Result	Actual Result	Status
TC-014	Load Grading Form	Select Class: 1A, Subject: Matematika	List of students in Class 1A appears.	Student list loaded correctly.	**Valid**
TC-015	Input Numeric Grade	Input Score: 85	System auto-calculates Letter Grade "A".	Letter "A" appeared automatically.	**Valid**
TC-016	Edit Teacher Profile	Update Phone Number & Upload New Photo	Profile data updated in Sheet; Photo saved to Drive.	Profile updated successfully.	Valid
TC-017	Save Grades	Click "Simpan Nilai"	Grades written to "NilaiSiswa" Sheet.	Data persisted in Sheet.	**Valid**
TC-018	Upload Lesson Plan (RPP)	Upload PDF file "RPP_Minggu1.pdf"	File stored in Drive folder; link recorded.	File accessible in Drive.	**Valid**

4.3.1.4 Student Access and Notification Testing

The final stage of testing verified the accessibility of information for Students and Guardians. This included verifying that schedules and grade reports are displayed correctly according to the logged-in user's class. Additionally, the WhatsApp notification feature (via Wablas API) was tested to ensure broadcast messages were successfully delivered to parents. The results are presented in Table 4.4.

Table 4.4 Testing of Student Access and WhatsApp Notification

Test ID	Scenario	Test Input Data	Expected Result	Actual Result	Status
TC-019	View Schedule	Login as Student Class 1A	Schedule for Class 1A displayed.	Correct schedule displayed.	**Valid**
TC-020	View Grades	Select Year: 2025/2026, Semester: 1	Grade report table rendered.	Report card displayed.	**Valid**
TC-021	Edit Profile (Allowed)	Change Phone Number	Phone number updates in database.	Update successful.	**Valid**
TC-022	Edit Profile (Restricted)	Try to change NISN or Name	Field is read-only (cannot be edited).	Field locked/grayed out.	**Valid**
TC-023	WhatsApp Welcome Msg	Admin register new student	WA message sent to parent number.	Message received on WA.	**Valid**
TC-024	WhatsApp Broadcast	Admin sends "Libur Sekolah" to Class 1A	All parents in Class 1A receive message.	All 30 parents received msg.	**Valid**

4.3.2 Usability Evaluation (SUS)

To evaluate the system's acceptance, a System Usability Scale (SUS) test was conducted. The survey involved 96 respondents from SD Muhammadiyah Kadisoka, comprising:

- 36 Internal Staff (Principal, Teachers, Administrative Staff).
- 60 External Users (Student Guardians).

This diverse demographic ensures the evaluation captures the perspective of both the technical operators (staff) and the end-users (guardians). The result is in Tabel 4.1.

Tabel 4.1 Summary of SUS Scores by User Group

User Group	Respondents (N)	Average SUS Score	Category
Teachers & Staff	36	88.4	Excellent
Student Guardians	60	78.1	Good
Overall Average	96	81.9	Excellent

The SUS results indicate a clear difference in satisfaction levels between user groups. Teachers and staff achieved the highest average score (88.4), placing them in the Excellent category. This suggests that the system effectively addresses their daily administrative needs and significantly reduces previous workload issues. A key contributing factor to this high satisfaction is the availability of digital teaching plan uploads, which eliminates reliance on

physical documents. Feedback from several teachers, including Muntasiyah and Fimpi Kurnia, shows that the interface is intuitive and well aligned with routine teaching tasks.

Student guardians recorded a slightly lower average score (78.1), which still falls within the Good category. This indicates overall acceptance of the system, though with a modest learning curve. Some guardians required additional time to understand certain features, particularly the need to manually select the academic year when viewing grade reports. Despite this, many guardians expressed positive perceptions, highlighting the convenience of accessing schedules and academic results remotely without visiting the school. Overall, the findings demonstrate that the system is highly usable across user groups, with particularly strong acceptance among teachers.

4.4 Discussion

The results of system implementation and testing confirm that a serverless, cloud-based architecture is an effective solution for addressing the operational constraints faced by SD Muhammadiyah Kadisoka. By eliminating the need for on-premise servers and dedicated IT personnel, the system aligns well with the school's limited financial and technical capacity. This finding supports the feasibility of using lightweight cloud services to deliver practical school administration functions without high infrastructure costs.

One significant finding from the usability evaluation is the difference between the System Usability Scale (SUS) scores of teachers/staff and student guardians. Teachers and staff achieved an average score of 88.4 (Excellent), indicating strong acceptance and ease of use. This outcome can be attributed to the system's alignment with their daily workflows, particularly features such as digital lesson-plan submission and centralized grade management, which directly replace time-consuming manual processes. In contrast, student guardians recorded a slightly lower, yet still positive, average SUS score of 78.1 (Good). This difference highlights an important usability insight: while the system is functionally effective, non-technical users may require additional interface simplification. For example, the need to manually select academic years and semesters when viewing grades introduces a minor cognitive burden. This finding emphasizes that usability perception is influenced not only by system functionality but also by user familiarity with digital systems.

From an architectural perspective, the use of Google Sheets as a database demonstrates that spreadsheet-based storage can function reliably for academic administration in environments with predictable data volume and low concurrency. The system successfully

supports core data entities such as student profiles, teacher records, schedules, grades, and document links in a format that remains accessible for non-technical staff. However, the implementation also reveals several platform limitations that must be acknowledged. Since Google Sheets is not a transactional database, it does not provide strict schema enforcement or advanced concurrency control. Under conditions of heavy simultaneous write operations, there is a risk of data collision or conflicting updates if multiple users modify records at the same time. Additionally, because the backend runs on Google Apps Script, long-running operations are constrained by execution timeouts and quota limits, meaning large-scale automated processes must be designed carefully to avoid forced termination.

The integration of WhatsApp notifications via the Wablas API also proved to be a critical success factor in extending the system's value beyond internal administration. Despite inherent delivery delays caused by rate limiting and anti-spam safeguards, the notification system effectively bridges the communication gap between the school and parents. This capability enhances parental engagement by enabling the school to deliver timely announcements directly through a familiar platform, which is not supported by the national Dapodik system. The broadcast mechanism, while introducing delivery latency for larger target groups, improves reliability by ensuring messages are sent sequentially in a controlled manner and reducing the likelihood of blocked requests.

Beyond technical feasibility, the primary value of the developed system lies in its ability to automate routine administrative and academic processes, thereby reducing manual workload and improving operational efficiency for all user groups. The automation features were intentionally designed to align with existing school workflows, minimizing disruption while replacing repetitive tasks with system-driven processes. For administrators, automation is most evident in student and teacher registration workflows. When a new student record is created, the system automatically generates a unique identifier, stores profile data, uploads supporting documents to Google Drive, and triggers a personalized WhatsApp welcome message containing login credentials. This end-to-end automation eliminates several manual steps previously handled through paper forms, phone calls, or in-person coordination.

For teachers, the system automates academic grading and document submission. Numeric grades entered through the interface are automatically converted into standardized letter grades, ensuring consistency across subjects and reducing calculation errors. The duplicate-prevention logic embedded in the grading module further improves data quality by ensuring that grade records are updated correctly without creating redundant rows for the same

student, semester, and academic year. In addition, the digital upload of lesson plans (RPP) replaces physical document submission, simplifying archival processes and ensuring that teaching documents remain traceable and accessible through a centralized platform.

For students and guardians, automation manifests as self-service access to academic information. Class schedules and grade reports are dynamically retrieved based on authenticated user credentials, enabling guardians to monitor academic progress without depending on manual reporting or in-person visits. This directly addresses the transparency gap identified during the preliminary study, where parents previously relied on indirect communication channels or limited access to academic records. Even though guardians required minor adaptation to interface conventions such as selecting academic year and semester, their overall usability score indicates positive acceptance of the system's benefits.

Overall, the findings confirm that the system's impact is not driven by technological complexity, but by purposeful automation of high-frequency tasks using appropriate and sustainable cloud tools. While the architecture is subject to constraints such as concurrency sensitivity and Apps Script timeouts, these limitations did not become operational barriers within the scale of SD Muhammadiyah Kadisoka. Therefore, the developed MIS demonstrates that a low-cost, serverless approach can provide meaningful improvements in school administration, teacher workflow efficiency, and parent engagement when implemented within realistic institutional boundaries.

CHAPTER V
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This research successfully designed, developed, and validated a serverless, zero-cost Management Information System tailored to the operational constraints and specific needs of SD Muhammadiyah Kadisoka. The system effectively bridges the critical digital gap between mandatory external reporting via the national Dapodik platform and the school's internal, day-to-day administrative needs. By leveraging the Google Workspace ecosystem—Google Apps Script as the backend runtime, Google Sheets as the primary datastore, and Google Drive for document storage—the architecture eliminated the need for local server hardware, upfront investment, and ongoing maintenance costs. This fully achieved the first objective, validating the proposed technical stack as a practical and sustainable alternative for schools with limited financial and technical capacity.

The implementation of role-based functionality successfully addressed the school's requirement for segmented digital access, fulfilling the second research objective. Administrators gained full CRUD control over all master data—including students, teachers, and classes—along with the ability to configure academic schedules and send broadcast messages. Teachers were provided with a dedicated digital platform to upload lesson plans directly to Google Drive, replacing the physical submission process, and to input student grades with automatic conversion from numerical scores to letter grades. Furthermore, students and guardians were granted essential self-service access to view real-time academic schedules and grade reports, a capability previously unavailable, along with limited ability to update personal contact information. All core functionalities for each role were implemented and verified as operational through comprehensive black-box testing.

Integrating a WhatsApp notification gateway, was successfully realized through the Wablas API. This integration enables the system to automatically send welcome messages with login credentials upon new student registration and allows administrators to broadcast announcements directly to parents. By leveraging Indonesia's most ubiquitous communication channel, this feature significantly enhances the school's ability to push critical information to guardians, thereby closing a longstanding communication gap and moving beyond the limitations of the admin-only Dapodik platform.

The system underwent rigorous validation through functional and usability testing, satisfying the fourth objective. Black-box testing confirmed that all specified functional requirements operated correctly, with core processes for data management, grading, document upload, and schedule viewing performing as expected. The System Usability Scale evaluation, conducted with 96 respondents from the school community, yielded an overall excellent average score of 81.9, significantly exceeding the accepted threshold of 68 and indicating high user acceptance. A notable finding was the divergence between user groups: teaching staff awarded an excellent score of 88.4, reflecting the system's strong alignment with their workflow, while student guardians gave a lower but still good score of 78.1, indicating general acceptance alongside an identified need for further interface simplification.

However, it is important to acknowledge that the chosen "zero-cost" architecture introduces specific technical limitations. As the system relies on Google Sheets rather than a transactional database, it is not designed for high-frequency concurrent usage; simultaneous write operations by more than 30–50 users may result in "row-locking" or performance degradation. Additionally, the backend logic is bound by Google Apps Script's execution timeouts (approximately 6 minutes), which constrains the ability to process extremely large datasets or send bulk WhatsApp broadcasts instantaneously. These limitations are acceptable trade-offs for a low-cost school environment but define the boundaries of the system's scalability compared to enterprise SQL solutions.

In summary, the developed MIS has successfully transformed administrative and academic management at SD Muhammadiyah Kadisoka from fragmented, paper-based workflows into an integrated and cloud-accessible digital platform. Through role-based access, automated data management, digital document submission, and improved school–parent communication via WhatsApp notifications, the system provides measurable operational benefits for administrators, teachers, and guardians. Although the Google Apps Script and Google Sheets architecture introduces scalability constraints such as execution time limits and sensitivity to high concurrent write activity, these trade-offs remain acceptable within the school's low-to-moderate usage context. This research confirms that a carefully designed serverless approach using widely available, free tools can serve as a practical and sustainable solution to reduce the digital divide in resource-constrained religious and community-based schools in Indonesia.

5.2 Contribution Statement

This research contributes to both academic knowledge and practical implementation in the field of educational information systems, particularly for resource-constrained institutions.

From a technical contribution perspective, this study demonstrates that a fully serverless architecture using Google Apps Script, Google Sheets, and Google Drive can function as a reliable backend for a multi-role School Management Information System. The research validates the feasibility of using spreadsheet-based storage for structured academic data when system scope and data volume are clearly defined.

From a practical contribution, this research delivers a functional, zero-cost MIS prototype that directly addresses the real operational challenges of SD Muhammadiyah Kadisoka. The system improves administrative efficiency, reduces dependency on physical documents, enhances transparency for parents, and strengthens school–parent communication through automated WhatsApp notifications.

From a methodological contribution, this study provides a documented case study of applying the Waterfall development model in an educational environment with fixed and well-defined requirements. The combined use of Black-box testing and the System Usability Scale (SUS) offers a replicable evaluation framework for similar academic system development projects.

Collectively, these contributions support the argument that meaningful digital transformation in education does not necessarily require complex infrastructure or high financial investment, but rather appropriate technological choices aligned with institutional context.

5.3 Recommendations

Based on the implementation experience, testing outcomes, and user feedback, the following recommendations are proposed for improving the current system and guiding future research:

a. Enhance User Experience for Guardians: To address the usability gap identified by the lower SUS scores among guardians, the student/guardian module should be optimized, especially the system user interface.
b. Optimize Notification Performance: The current sequential message sending with a 20-30 second delay, while functional, is not optimal for large-scale broadcasts. Future work should implement an asynchronous message queuing system within the backend logic.

This would allow the administrator to initiate a broadcast and have the system process the queue in the background, improving the user experience and potentially leveraging more efficient API call patterns if the service quota is upgraded.

c. Decouple Application Logic from Data Structure: The black-box testing revealed a bug where hiding columns in the Google Sheets interface broke the web application's table rendering. This is due to the code's reliance on fixed column indices. It is recommended to refactor the data retrieval functions to dynamically map column headers, making the frontend resilient to changes in the backend spreadsheet layout.

d. Expand Module Functionality: As the school becomes more accustomed to the digital system, future development can focus on adding modules identified as beneficial but outside the initial scope. These could include a library management system, or a simple inventory module for school assets, all built within the same serverless paradigm.

e. Conduct Longitudinal Study: A valuable avenue for further research would be a longitudinal study assessing the system's impact on administrative efficiency at SD Muhammadiyah Kadisoka over one or two academic years. Metrics could include time saved on report generation, reduction in data entry errors, and trends in parent engagement through the notification system. This would provide empirical evidence of the long-term benefits of such digital interventions.

f. Develop a Generalizable Framework: This case study provides a blueprint. Future research could focus on abstracting the core architecture and codebase to create a configurable template or framework. This would allow other schools with similar constraints to deploy their own tailored version of the system with minimal coding effort, amplifying the impact of this research.

g. Migration to Cloud SQL for High Concurrency: While Google Sheets serves the current population well, it is not suitable for high-concurrency environments. If the school expands significantly, it is recommended to migrate the database layer from Google Sheets to Google Cloud SQL or Firebase. This would resolve row-locking issues and eliminate the performance degradation associated with large spreadsheet datasets, while keeping the frontend interface largely intact.

REFERENCE

Abdugulova, Z. (2017). Allowing Schools Access to Affordable Computers. How Schools Can Benefit From Switching to Inexpensive, Cloud-Based Computing Technologies. *International Journal of Learning and Teaching*, *9*(3), 326–331. https://doi.org/10.18844/ijlt.v9i3.507

admin. (2021, June 3). How to Use Google Sheets as a Database and Its Various Use Cases? Retrieved December 24, 2025, from Smatbot—AI ChatBot Solutions website: https://www.smatbot.com/blog/google-sheets-as-a-database/

Andi Bulkis Magfirah Mannong. (2024). Google Drive as Instructional Media to Facilitate Students' Assignment Submission: Opportunities and Challenges. *INTERACTION: Jurnal Pendidikan Bahasa*, *11*(2), 560–569. https://doi.org/10.36232/interactionjournal.v11i2.278

Bassil, Y. (2012, May 31). *A Simulation Model for the Waterfall Software Development Life Cycle*. arXiv. https://doi.org/10.48550/arXiv.1205.6904

Bendre, M., Sun, B., Zhang, D., Zhou, X., Chang, K. C., & Parameswaran, A. (2015). DataSpread: Unifying Databases and Spreadsheets. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, *8*(12), 2000–2003.

Bendre, M., Venkataraman, V., Zhou, X., Chang, K., & Parameswaran, A. (2018). Towards a Holistic Integration of Spreadsheets with Databases: A Scalable Storage Engine for Presentational Data Management. *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 113–124. Paris: IEEE. https://doi.org/10.1109/ICDE.2018.00020

Clark, N., Dabkowski, M., Driscoll, P., Kennedy, D., Kloo, I., & Shi, H. (2021). *Empirical Decision Rules for Improving the Uncertainty Reporting of Small Sample System Usability Scale Scores*. https://doi.org/10.48550/ARXIV.2101.00455

Dedi Jubaedi, A., Dwiyatno, S., Krisnaningsih, E., Solihin, Shafitri, A., & Sutiawan, A. (2023). SISTEM INFORMASI MONITORING KEGIATAN ABSENSI SISWA DENGAN NOTIFIKASI WHATSAPP. *JSiI (Jurnal Sistem Informasi)*, *10*(2), 109–115. https://doi.org/10.30656/jsii.v10i2.6630

Ferrell, N. (2018, September 10). Going Serverless for free with Google Apps Script + Google Sheets. Retrieved December 24, 2025, from Ferrell.io website: https://ferrell.io/blog/2018-09-10-appsscript-free-serverless/

Frayudha, A. D., Pande, I. R., & Juwita, M. B. (2024). Implementation of Black Box Testing with the Application of Equivalence Partitioning Techniques in the M-Magazine Android Application at Semen Gresik High School. *Elinvo (Electronics, Informatics, and Vocational Education)*, *9*(1), 134–143. https://doi.org/10.21831/elinvo.v9i1.70382

Gandhi, S., Gore, A., Nimbarte, S., & Abraham, J. (2018). Implementation and Analysis of a Serverless Shared Drive with AWS Lambda. *2018 4th International Conference for Convergence in Technology (I2CT)*, 1–6. Mangalore, India: IEEE. https://doi.org/10.1109/I2CT42659.2018.9058237

Ghorbian, M., & Ghobaei-Arani, M. (2025). *Serverless Computing: Architecture, Concepts, and Applications* (Version 1). Version 1. arXiv. https://doi.org/10.48550/ARXIV.2501.09831

Google Apps Script overview. (n.d.). Retrieved December 24, 2025, from Google for Developers website: https://developers.google.com/apps-script/overview

Gupron, G., Yandi, A., Suprapto, E., & Sadewa, I. (2024). Management Information System (MIS) for Achieving Work Efficiency and Effectiveness in Maximizing Employee Performance: A Conceptual Study as a Guide for Researchers. *Siber International Journal of Digital Business (SIJDB)*, *2*(2), 149–162. https://doi.org/10.38035/sijdb.v2i2.135

Hasim Iswanto, M. (2024). Design of Web-based Report Card Processing Information System at MTS Negeri 4 Kebumen. *Jurnal Indonesia Sosial Teknologi*, *5*(5), 2184–2191. https://doi.org/10.59141/jist.v5i5.1047

Inda Fathya, N. 21210054. (2025). *SISTEM INFORMASI KEHADIRAN SISWA BERBASIS WEB MENGGUNAKAN API WHATSAPP DI SMK NEGERI 1 AL MUBARKEYA ACEH BESAR* (Diploma, Universitas Bina Bangsa Getsempena). Universitas Bina Bangsa Getsempena. Retrieved from https://library.bbg.ac.id/

Insan Asry, A. (2022). Implementation of Google App Script in Cloud-Based Data Search Application. *JEAT : Journal of Electrical and Automation Technology*, *1*(2), 88–93. https://doi.org/10.61844/jeat.v1i2.405

Kasbijanto, K., Mohammad Syahidul Haq, Amalia, K., & Hazin, M. (2025). Policy Analysis of the Integration between Dapodik and the Online Student Admission System in Bojonegoro Regency. *International Journal of Educational Evaluation and Policy Analysis*, *2*(4), 69–77. https://doi.org/10.62951/ijeepa.v2i4.407

Liu, W., & Zeng, Q. (2024). Hybrid Cloud Computing: An In-Depth Analysis of Integration Strategies, Characteristics, and Prospective Future Applications. *Innovation in Science and Technology*, *3*(1), 10–13.

Mandei, J. M., Siang, J. L., & Mamahit, M. M. (2025). The Effectiveness of the Management Information System in Improving the Quality of Education at SMP Negeri 1 Tumpaan, South Minahasa Regency. *International Journal of Education, Information Technology, and Others*, *8*(3.B), 211–218.

Moss, A. (2016). Rising Above the Clouds: A Review of the Implications that Cloud Computing Technologies Hold for Education. *Journal of Cloud Computing*, 1–13. https://doi.org/10.5171/2016.623649

Murphy, S. (2008). *Comparison of Spreadsheets with other Development Tools (limitations, solutions, workarounds and alternatives)*. https://doi.org/10.48550/ARXIV.0801.3853

Ostrand, T. (2002). Black-Box Testing. In J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering* (1st ed.). Wiley. https://doi.org/10.1002/0471028959.sof022

Pollard, B. (2024, March 22). How Education Benefits From Utilizing Cloud Storage. Retrieved January 5, 2026, from https://adivi.com/blog/how-education-benefits-from-utilizing-cloud-storage/

Pratama, D. A., & Hindarto, H. (2025, March 6). *Whatsapp Integrated Information System on Student Attendance Management: Sistem Informasi Terintegrasi Whatsapp pada Manajemen Absensi Siswa*. https://doi.org/10.21070/ups.7557

Rodiyah, I., Rustianingsih, E., & Solicha, F. (2024). Optimizing Dapodik Data Management in the Digital Era: Case Study of SMA Negeri 1 Krian. *Indonesian Journal of Public Administration Review*, *1*(4), 8. https://doi.org/10.47134/par.v1i4.3891

Rosanti, R. L., & Swalaganata, G. (2024). Implementasi Google App Script untuk Data Entry pada Master Data. *REMIK: Riset Dan E-Jurnal Manajemen Informatika Komputer*, *8*(1), 117–129. https://doi.org/10.33395/remik.v8i1.13273

S, Y. A., Ernawati, S., Saputra, H., & Kurniawan, M. A. (2024). Islamic Education Management Strategy in the Digital Era: Governance Transformation to Increase Effectiveness and Accessibility. *International Journal of Islamic Educational Research*, *1*(4), 27–44. https://doi.org/10.61132/ijier.v1i4.67

Sander, A. (2025, February 20). Google Workspace for Education: A Guide for K-12 Schools. Retrieved December 24, 2025, from ManagedMethods Cybersecurity, Safety & Compliance for K-12 website: https://managedmethods.com/blog/g-suite-for-education/

Sasmito, G. W., & Mutasodirin, M. A. (2023). Black Box Testing with Equivalence Partitions Techniques in Transcrop Applications. *2023 6th International Conference of Computer and Informatics Engineering (IC2IE)*, 53–58. Lombok, Indonesia: IEEE. https://doi.org/10.1109/IC2IE60547.2023.10331562

Shanganlall, A. (2024, June 25). Cloud-Based Student Information Systems for K12 Schools. Retrieved December 24, 2025, from Classter website: https://www.classter.com/blog/edtech/cloud-based-student-information-systems-for-k12-schools/

Siskawati, I., Masturi, A., Hidayati, N., Sary, M. P., & Nisoh, A. (2025). The Influence of Whatsapp Usage and Teacher Interpersonal Communication on Organizational Communication Quality. *Syiar: Jurnal Komunikasi Dan Penyiaran Islam*, *5*(2), 269–280. https://doi.org/10.54150/syiar.v5i2.763

Surahman, S. (2023, November 2). *Unveiling the Determinants of Actual Use among, Google Drive Users: A Comprehensive Analysis* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. https://doi.org/10.2139/ssrn.4620933

System Usability Scale—An overview | ScienceDirect Topics. (n.d.). Retrieved December 24, 2025, from https://www.sciencedirect.com/topics/computer-science/system-usability-scale

Thorne, S., & Hancock, J. (2019). *A Case Study of Spreadsheet Use within the Finance and Academic Registry units within a Higher Education Institution*. https://doi.org/10.48550/ARXIV.1909.07462

University of Novi Sad - Faculty of Economics, Segedinski put 9-11, 24000 Subotica, Matković, P., Tumbas, P., & University of Novi Sad - Faculty of Economics, Segedinski put 9-11, 24000 Subotica. (2010). A Comparative Overview of the Evolution of Software Development Models. *International Journal of Industrial*

Engineering and Management, *1*(4), 163–172. https://doi.org/10.24867/IJIEM-2010-4-019

Veuvolu, R., Suryadevar, A., Vignesh, T., & Avthu, N. R. (2023). Cloud Computing Based (Serverless computing) using Serverless architecture for Dynamic Web Hosting and cost Optimization. *2023 International Conference on Computer Communication and Informatics (ICCCI)*, 1–6. Coimbatore, India: IEEE. https://doi.org/10.1109/ICCCI56745.2023.10128286

Weis, O. (2024, April 5). Amazon S3 vs Google Cloud Storage: What are the differences? Retrieved January 5, 2026, from Cloud storage manager – Manage multiply cloud services website: https://cloudmounter.net/amazon-s3-vs-google-cloud-storage/

APPENDIX

Appendix, Socialization, Data Gathering and Testing to Sd Muhammadiyah Kadisoka