Journal of Computer Science and Technology
Volume 4 ; Number 3 ; September 2026 ; Page 248-256
DOI : https://doi.org/10.59435/jocstec.v4i3.865
https://jurnal.padangtekno.com/index.php/jocstec
E-ISSN : 2985-4318
P-ISSN : 2985-5772

Development Of A Web-Based Course Scheduling Information System For The Information Technology Study Program At Universitas PGRI Sumatera Barat

M Egi Alfarizi[1], Ade Pratama[2], Rahayu Trisetyowati Untari[3]

[1-3]Information Technology Study Program, Faculty of Science and Technology, Universitas PGRI Sumatera Barat
[1] halo.megialfarizi@gmail.com, [2] adepratama984@gmail.com, [3] 3.untari@gmail.com

ABSTRACT

Course timetable preparation in the Information Technology Study Program at Universitas PGRI Sumatera Barat is still performed in spreadsheets, typically requires three to five days, and can produce instructor, class, or room conflicts. This study develops a web-based scheduling information system that centralizes timetable data, detects collisions automatically, and offers alternatives with support from the Google Gemini API. Development followed the Waterfall System Development Life Cycle, comprising requirements analysis, design, implementation, testing, deployment, and maintenance. The application was built with PHP, Laravel, and MySQL. A deterministic rule identifies overlapping allocations involving the same instructor, class, or room on the same day. Gemini is used only to rank and explain feasible candidates; the application revalidates each recommendation, and the Study Program Secretary retains final approval. Black-box testing of ten functions achieved a 100% success rate. Beta testing produced an average score of 89.22% from three experts and 89.79% from two users, with both results classified as Very Good. The findings indicate that the system is highly feasible for integrated course timetable preparation and administration.

Keywords: course scheduling, web application, Google Gemini API, Waterfall, conflict detection

INTRODUCTION

Digital transformation in higher education extends beyond delivering academic information; it also requires institutions to redesign workflows that once relied on disconnected documents. Course timetabling is particularly demanding because it must coordinate instructors, courses, class groups, rooms, credit loads, days, and time slots. In a manual workflow, even a minor adjustment to one element can trigger changes across several other entries. Prior work on web-based scheduling indicates that consolidated data can speed up information retrieval, improve record accuracy, and reduce repetitive administrative tasks [1], [2].

The Information Technology Study Program at Universitas PGRI Sumatera Barat currently prepares its timetable in spreadsheets. Courses are entered individually by the Study Program Secretary, who then checks instructor, class, and room conflicts by hand. Observations and interviews showed that completing this process generally takes three to five days. Moreover, instructor, course, class, room, academic-year, and finalized-schedule records are not yet connected through a single platform. This fragmented arrangement makes data-entry errors, duplicate allocation of resources, and delays during schedule revisions more likely.

Several studies have addressed web-based course scheduling. Using the Waterfall model, Bunduwula et al. created a system that presents timetables, assigned instructors, and course-delivery status [3]. Fitria and Nunsina likewise reported that a web application made schedule storage and presentation faster and more practical [2]. These contributions, however, primarily concerned

schedule administration and dissemination; neither placed integrated conflict checking with alternative recommendations at the center of the workflow.

Optimization techniques have also been applied to reduce timetable collisions. Ihtiara and Abdul Muthalib used a Greedy algorithm that incorporated instructor preferences [4], while Nuradira et al. adopted a genetic algorithm to improve the use of time slots and rooms [5]. Although such techniques can produce favorable schedule combinations, their implementation depends on problem-specific constraints and optimization procedures. In the setting examined here, the immediate requirement is not a system that completely replaces managerial judgment. Instead, users need reliable rule-based conflict checks, feasible alternatives, and continued authority for the Study Program Secretary to approve the final timetable.

Other work has applied Kanban to visualize a scheduling process previously managed in Excel [6], and real-time web scheduling has been found to accelerate the communication of timetable changes [7]. Workflow visibility and rapid distribution alone are insufficient, however, when the system cannot verify whether an instructor, class, or room has been assigned to overlapping periods. Consequently, the proposed application treats deterministic conflict validation as a prerequisite for saving any schedule entry.

Recent advances in large language models create opportunities for decision support in academic systems. Google Gemini can interpret structured inputs and generate context-sensitive explanations or recommendations [8]. Nevertheless, language models have recognized limitations in planning and should not be treated as authoritative without independent verification [9], [10]. For this reason, Gemini does not approve schedules in the proposed system. The application first generates available combinations of days, times, and rooms, sends these candidates as JSON, and then asks Gemini to rank and explain them. Before any option is shown to the user, it is checked again against the system's conflict rules.

The research gap concerns the limited integration of centralized scheduling data, deterministic conflict detection, AI-assisted alternatives, post-recommendation validation, and human approval within one process. This study therefore develops a Web-Based Course Scheduling Information System for the Information Technology Study Program at Universitas PGRI Sumatera Barat. The intended system reduces manual effort, lowers the likelihood of timetable conflicts, supports faster revisions, and provides finalized schedules that authorized users can view, print, or download.

RESEARCH METHODOLOGY

Research Setting, Approach, and Data Collection

The study was conducted in the Information Technology Study Program at Universitas PGRI Sumatera Barat during the second semester of the 2026/2027 academic year. A development-research approach was followed through the Waterfall form of the System Development Life Cycle (SDLC). This model was selected because its sequential structure allows user requirements to be established before design, coding, and testing begin [11].

Requirements were gathered by observing the existing timetable-preparation process and interviewing the Study Program Secretary, who is responsible for schedule administration. The observation examined data-entry procedures, conflict-checking practices, revision activities, and the distribution of the final timetable. The interview explored functional needs, access privileges, mandatory records, and expected outputs. Evidence from both methods informed the process model and database design.

System Development Stages

Development proceeded through requirements analysis, design, implementation, testing, deployment, and maintenance. Figure 1 illustrates how these stages are connected. Each phase produced an output for the next phase, while findings identified during testing were used for limited corrections to the implementation.

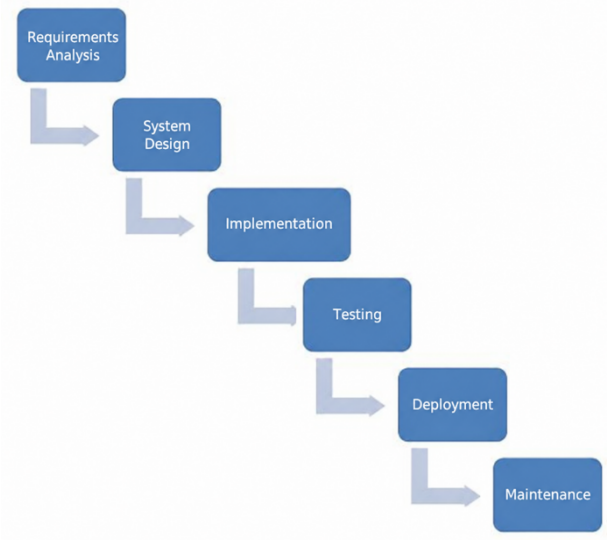

Figure 1. System development stages using the Waterfall model

Requirements analysis. Two primary actors were identified: the Study Program Administrator and the Study Program Secretary. Functional requirements covered authentication; management of instructor, course, class, room, and academic-year data; timetable preparation; conflict detection; alternative recommendations; final schedule approval; and schedule printing or downloading. Nonfunctional requirements addressed access security, performance, usability, availability, reliability, and portability.

Design. Unified Modeling Language was used to represent actor relationships, activity flows, interaction sequences, and class structures [16]. The database was designed so that every schedule record consistently references the relevant instructor, course, class, room, and academic year. The design also preserved the Study Program Secretary's authority over final decisions.

Implementation. The application was developed in PHP with Laravel and a MySQL database. Laravel supported separation of application logic, routing, authentication, and data access [12], whereas MySQL provided centralized storage for master records and scheduling transactions [13]. Google Gemini was accessed through an API with structured inputs and outputs, allowing recommendations to be parsed and reprocessed by the application [15].

Testing and deployment. Alpha evaluation combined white-box testing of core logic with black-box testing of functional outputs. Once the functions operated as intended, experts and end users participated in beta testing. The system was then prepared for use within the Study Program, together with plans for maintenance, defect correction, and future feature development.

Table 1. System actors and access privileges

No.	Actor	Primary Access Privileges
1	Study Program Administrator	View, search, print, and download schedules that have been approved as final.
2	Study Program Secretary	Manage master data, prepare schedules, detect conflicts, assess alternative recommendations, and approve the final timetable.

Conflict Detection and Recommendation Mechanism

Whenever a timetable entry is created or modified, the application compares it with stored records. A conflict is registered when the same instructor, class, or room is assigned on the same day and the two time intervals overlap. Equation (1) expresses the interval-overlap rule.

$$\text{Conflict} = (\text{same entity}) \text{ AND } (\text{day1} = \text{day2}) \text{ AND } (\text{start1} < \text{end2}) \text{ AND } (\text{start2} < \text{end1}) \quad (1)$$

For transparency, a simplified representation of the exchange is used: the Laravel request contains {"conflicts":[{"entity":"instructor|class|room","id":"<id>","day":"<day>","start":"<time>","end":"<time>"}], "candidates":[{"day":"<day>","start":"<time>","end":"<time>","room_id":"<id>"}]}; the response is returned as {"ranked_candidates":[{"day":"<day>","start":"<time>","end":"<time>","room_id":"<id>","reason":"<explanation>"}]}. The system instruction restricts Gemini to the supplied candidates, and Laravel parses and validates these fields again before display or storage.

An entry with no detected conflict may be saved. When a collision occurs, the application searches for unused combinations of day, time, and room and sends the resulting candidates to Google Gemini as JSON. Gemini ranks the options and supplies an explanation, but its response does not constitute an official schedule. The application revalidates every suggestion with the same deterministic rule, after which the Study Program Secretary chooses the most appropriate alternative. This arrangement limits the effect of inconsistent AI output while retaining human control over the final decision.

White-box testing covered authentication, master-data administration, timetable construction, conflict validation, alternative generation, and delivery of the finalized schedule. Cyclomatic Complexity determined which independent execution paths required testing. Black-box testing then evaluated ten core functions by comparing their inputs and outputs without examining the source code [14].

Beta testing employed a four-point Likert questionnaire. Three experts assessed functionality, reliability, usability, efficiency, and maintainability. Two system users evaluated format, accuracy, content, ease of use, and timeliness. The two users were purposively selected because they were the operational staff directly involved in timetable preparation and schedule administration. Feasibility percentages were obtained by dividing each observed score by its ideal score and multiplying the result by 100%. Scores from 81% to 100% were classified as Very Good.

RESULTS AND DISCUSSION

System Implementation

The prototype interface is presented in Indonesian because the system is intended for operational use by the local Study Program staff; the article text and technical labels remain in English.

The resulting web-based course scheduling system combines authentication, role-specific dashboards, master-data management for instructors, courses, classes, rooms, and academic years, schedule construction, conflict detection, alternative recommendations, final schedule approval, and print/download facilities. MySQL serves as the shared data source, so every revision is managed through the same centralized repository.

The Study Program Secretary is the principal actor in timetable preparation. Once the master records are available, the user selects a course, assigned instructor, class, room, day, start time, and end time. Before saving the entry, the application verifies that all required fields are complete and checks for possible conflicts. As shown in Figure 2, the scheduling page also supports search, data import, entry creation, and printing.

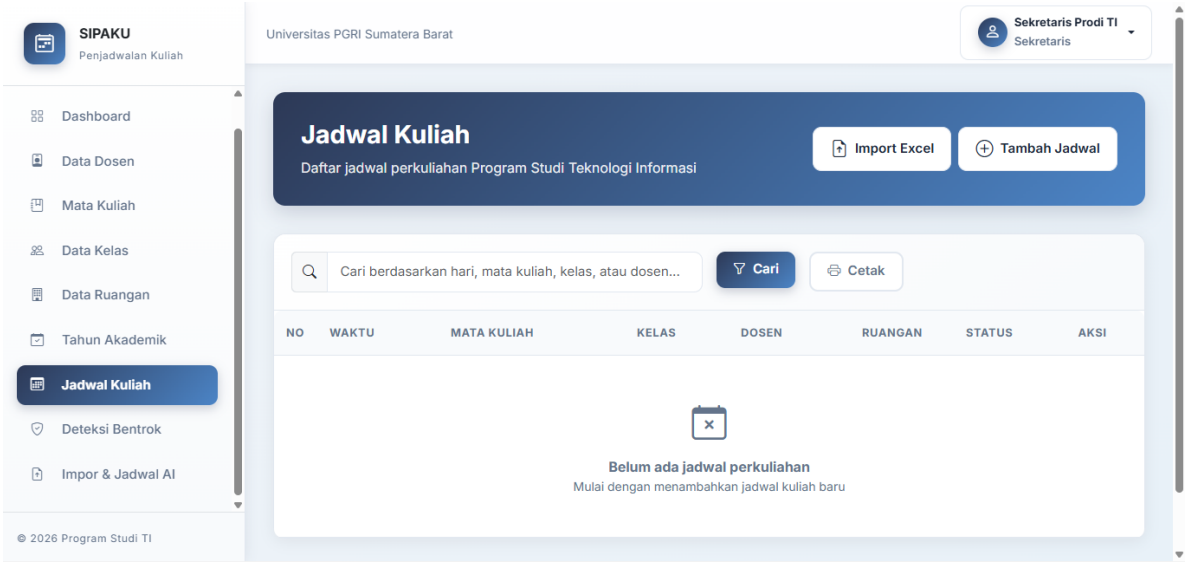

Figure 2. Course schedule management page

When the system detects a conflict, it blocks the entry and prepares feasible alternatives rather than merely rejecting it. The AI-integration module accepts either unorganized timetable records or a filtered candidate list and submits the data to Google Gemini. Figure 3 presents the module interface. AI output is first stored as recommendation data; only options that pass rule-based revalidation and receive user approval can be applied to the main timetable.

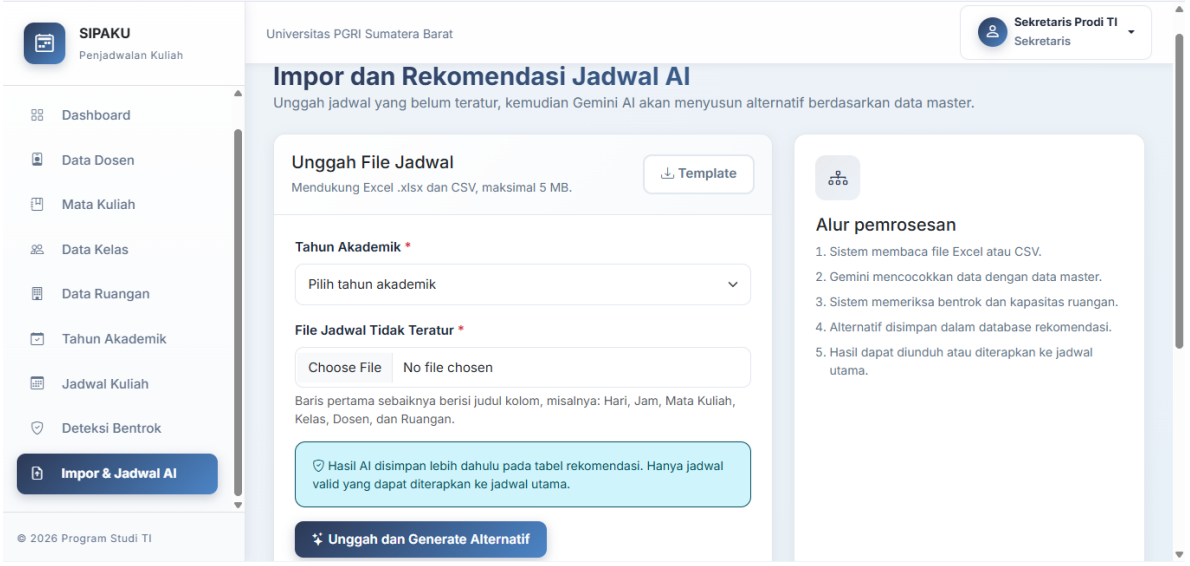

Figure 3. Google Gemini-assisted schedule import and recommendation page

Alpha Testing Results

White-box testing confirmed that the logical paths for authentication, master-data processing, schedule preparation, conflict checking, and alternative recommendation could be executed as designed. Ten principal functions were then assessed through black-box testing. As summarized in Table 2, every scenario returned the expected output.

Table 2. Summary of black-box testing results

No.	Function	Expected Outcome	Result
1	Multi-role login	A valid account is directed to the dashboard permitted for its role.	Valid

No.	Function	Expected Outcome	Result
2	Instructor data	Records can be added, modified, deleted, and displayed.	Valid
3	Course data	Course records can be managed and saved.	Valid
4	Class data	The latest class records are displayed correctly.	Valid
5	Room data	Room and capacity records can be managed.	Valid
6	Schedule preparation	Timetable entries can be prepared, saved, and displayed.	Valid
7	Conflict detection	Instructor, class, or room collisions are identified.	Valid
8	Alternative recommendation	Alternatives are displayed and the selected option can be saved.	Valid
9	Schedule printing	The final schedule can be viewed, printed, or downloaded.	Valid
10	Logout	The user's session is terminated securely.	Valid

All ten tested functions were valid, producing a success rate of 10/10 x 100% = 100%, which falls within the Very Good category. This outcome confirms that the core functions met the specified functional requirements. Functional success alone does not fully represent user experience, however; the evaluation therefore continued with beta testing.

Beta Testing Results

The three experts assigned an average score of 89.22%, categorized as Very Good. Maintainability received the highest score at 95.83%, whereas functionality had the lowest score at 85.00%. Table 3 provides the detailed expert assessment.

Table 3. Expert assessment results

Dimension	Percentage	Category
Functionality	85.00%	Very Good
Reliability	88.89%	Very Good
Usability	88.89%	Very Good
Efficiency	87.50%	Very Good
Maintainability	95.83%	Very Good
Average	89.22%	Very Good

The two users produced an overall mean of 89.79%, also in the Very Good category. Content achieved the highest score at 93.75%, indicating that the information matched operational scheduling needs. Ease of use was the lowest-rated dimension at 84.38%, although it remained within the Very Good range. Detailed user scores are shown in Table 4.

Table 4. User assessment results

Dimension	Percentage	Category
Format	91.67%	Very Good
Accuracy	91.67%	Very Good
Content	93.75%	Very Good
Ease of Use	84.38%	Very Good
Timeliness	87.50%	Very Good
Average	89.79%	Very Good

Discussion

Testing showed that the application consolidates functions that were previously distributed across several manual activities. This finding supports the results of Bunduwula et al. [3] and Fitria and Nunsina [2], who documented the value of web systems for timetable management and communication. The present system extends those earlier solutions by maintaining master records in a single database, screening conflicts before storage, offering alternative slots, and forwarding the approved timetable to the Study Program Administrator for printing or downloading.

Unlike the Greedy [4] and genetic-algorithm [5] approaches, this study does not attempt to compute a fully automated global optimum. Instead, it applies transparent rules to prevent instructors, classes, and rooms from being assigned to overlapping periods. This choice reflects the users' need to retain direct control over the finalized schedule. Google Gemini operates only after the candidates have been filtered and never substitutes for the scheduling rules. The model's ability to organize and explain alternatives can therefore be used without disregarding its known planning limitations [9], [10].

Structured output is essential to this integration. Conflict records and feasible alternatives are encoded as JSON, and Gemini's response is mapped back to day, time, and room attributes [15]. The application then performs another validation pass. The resulting decision process has three safeguards: deterministic rules establish feasibility, Gemini assists with ranking, and the Study Program Secretary authorizes the selection. Accordingly, the study's principal contribution is not the presence of an AI feature by itself, but the placement of AI within a controlled and auditable decision workflow.

A 100% black-box test result demonstrates that the designed functions operated correctly across the prepared test scenarios. The expert score of 89.22% indicates very strong technical acceptance, with maintainability evaluated most favorably. This result is consistent with the separation of master-data, scheduling, validation, and recommendation modules, which simplifies maintenance. The user score of 89.79% further indicates that the system's content, accuracy, and information format meet operational requirements.

Although every dimension was rated Very Good, the 84.38% score for ease of use identifies a clear opportunity for improvement. The workflow could be refined by simplifying schedule entry, adding contextual guidance, shortening conflict explanations, and reducing the number of actions required to select an alternative. These improvements matter because a scheduling system succeeds not only through correct logic, but also by enabling users to complete their work quickly and confidently.

The beta sample was limited to three experts and two users. The feasibility findings therefore describe implementation in the Information Technology Study Program and should not be generalized to faculties or universities with different scales and scheduling policies. Recommendation explanations also depend on internet access and the availability of Google Gemini. Even so, conflict detection and candidate presentation were designed to function without complete reliance on AI. Future development should consider integration with SIAKAD or SEVIMA, change notifications,

audit trails, data backups, instructor time preferences, room capacity and facilities, and testing with a larger user population.

CONCLUSION

A web-based course scheduling information system was successfully developed for the Information Technology Study Program at Universitas PGRI Sumatera Barat using the Waterfall SDLC, PHP, Laravel, and MySQL. The application unifies instructor, course, class, room, academic-year, and timetable data. Conflicts are detected when the same instructor, class, or room is allocated on the same day to overlapping time intervals. If a collision occurs, the system generates feasible candidates and uses Google Gemini to help rank and explain them. No recommendation becomes final until it has passed another rule-based validation and has been approved by the Study Program Secretary.

Black-box testing of ten functions achieved a 100% success rate. Beta evaluation yielded mean scores of 89.22% from three experts and 89.79% from two users, placing both results in the Very Good category. These findings indicate that the system is highly feasible for preparing, checking, approving, and distributing course schedules. Subsequent work should prioritize usability improvements, a broader respondent sample, integration with the university's academic systems, and a fallback mechanism for periods when Google Gemini is unavailable.

REFERENCES

[1] H. Paramata, A. Lahinta, and S. Suhada, "Perancangan Sistem Informasi Penjadwalan," *Diffusion J. Syst. Inf. Technol.*, vol. 2, no. 1, pp. 30-39, 2022.

[2] A. Fitria and N. Nunsina, "Perancangan Sistem Informasi Penjadwalan Kuliah Berbasis Web pada Fakultas Komputer dan Multimedia di UNIKI," *Device: J. Inf. Syst. Comput. Sci. Inf. Technol.*, vol. 3, no. 2, pp. 9-15, 2022, doi: 10.46576/device.v3i2.2696.

[3] T. B. Bunduwula, M. Nur Iksan, S. R. Syaifullah, and L. O. M. Fid Aksara, "Perancangan Website Sistem Informasi Jadwal Perkuliahan Jurusan Teknik Informatika Menggunakan Metode Waterfall," *Simkom*, vol. 9, no. 1, pp. 23-35, 2024, doi: 10.51717/simkom.v9i1.353.

[4] A. Ihtiara and M. Abdul Muthalib, "Penjadwalan Mata Kuliah Menggunakan Algoritma Greedy dengan Mempertimbangkan Preferensi Dosen," *J. Sains Teknol. 4.0*, vol. 1, no. 2, pp. 37-46, 2024.

[5] A. H. Nuradira, M. R. Anam, L. Amrita, M. Z. Izzati, and H. Saputro, "Optimalisasi Penjadwalan Kuliah Menggunakan Algoritma Genetika untuk Meningkatkan Efisiensi Jadwal pada Program Studi Sistem Informasi UNISNU Jepara," *J. Inf. Syst. Comput.*, vol. 4, no. 2, pp. 70-76, 2024, doi: 10.34001/jister.v4i2.1220.

[6] J. C. Joseph, A. Sujjada, K. Kamdan, and G. P. Insany, "Implementasi Metode Kanban pada Penjadwalan Mata Kuliah Berbasis Web," *J. Ekonomi Manajemen Sistem Informasi*, vol. 6, no. 5, pp. 3435-3445, 2025, doi: 10.38035/jemsi.v6i5.5298.

[7] A. D. Situmorang, S. R. N. Muhammad, and W. Ariyadi, "Pengembangan Sistem Penjadwalan Kuliah Real Time Berbasis Web," *J. Ilmu Teknik dan Komputer (JITKOM)*, vol. 8, no. 2, pp. 143-148, 2024, doi: 10.22441/jitkom.v8i2.010.

[8] M. Imran and N. Almusharraf, "Google Gemini as a Next Generation AI Educational Tool: A Review of Emerging Educational Technology," *Smart Learning Environments*, vol. 11, no. 1, 2024, doi: 10.1186/s40561-024-00310-z.

[9] A. Bonlarron, F. Regin, E. De Maria, and J. C. Regin, "Large Language Model Meets Constraint Propagation," in *Proc. IJCAI*, pp. 10036-10044, 2025, doi: 10.24963/ijcai.2025/1115.

[10] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati, "On the Planning Abilities of Large Language Models: A Critical Investigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 75993-76005, 2023.

[11] M. R. R. Hidayat, "Penerapan Metode System Development Life Cycle (SDLC) dalam Pembangunan Sistem Informasi Berbasis Komputer serta Relevansi UX dan UI," *J. Teknol. Sistem Informasi*, vol. 5, no. 2, pp. 145-151, 2025.

[12] F. Sinlae, E. Irwanda, Z. Maulana, and V. E. Syahputra, "Penggunaan Framework Laravel dalam Membangun Aplikasi Website Berbasis PHP," *J. Siber Multi Disiplin*, vol. 2, no. 2, pp. 119-132, 2024, doi: 10.38035/jsmd.v2i2.186.
[13] U. K. Siregar, T. A. Sitakar, S. Haramain, Z. N. S. Lubis, and U. Nadhirah, "Pengembangan Database Management System Menggunakan MySQL," *SAINTEK: J. Sains Teknol. Komput.*, vol. 1, no. 1, pp. 8-12, 2024.
[14] S. P. Ramadhani, A. F. Saputra, F. Dwiansyah, and I. Veritawati, "Pengujian Sistem Informasi Akademik (NeoSiak) Berbasis Website Menggunakan Equivalence Partitioning dan Metode Black Box," *STORAGE: J. Ilm. Tek. Ilmu Komput.*, vol. 3, no. 1, p. 18, 2024.
[15] Google AI for Developers, "Structured outputs," Google, 2026. [Online]. Available: https://ai.google.dev/gemini-api/docs/structured-output. [Accessed: Aug. 12, 2026].
[16] T. A. Rospricilia and M. N. P. Ma'ady, "Pemodelan Integration Use Case (IUC): Perancangan Use Case Diagram (UML) untuk Sistem-Sistem yang Terintegrasi," *INTEGER: J. Inf. Technol.*, vol. 9, no. 2, pp. 165-172, 2024, doi: 10.31284/j.integer.2024.v9i2.6345.