MJI
Media Jurnal Informatika
Vol. 17, No. 2, December 2025
p-issn : 2088-2114, e-issn : 2477-2542

A Solution Recommendation System Based on Application Constraint History Using Cosine Similarity and Gemini AI

Ahmad Sidik Rudini [a,1,*], Novi Dian Nathasia [b,2], Cian Ramadhona Hassolthine [c,3], Anang Aris Widodo [d,4]

[a] Program Studi PJJ Informatika, Universitas Siber Asia, Jalan R.M. Harsono No. 1, RT 09/RW 04, Ragunan, Pasar Minggu, Jakarta Selatan 12550, DKI Jakarta, Indonesia
[b] Program Studi Sistem Informasi, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional, Jl. Sawo Manila No.61, Pejaten, Pasar Minggu, Jakarta Selatan 12520, DKI Jakarta, Indonesia
[c] Program Studi PJJ Informatika, Universitas Siber Asia, Jalan R.M. Harsono No. 1, RT 09/RW 04, Ragunan, Pasar Minggu, Jakarta Selatan 12550, DKI Jakarta, Indonesia
[d] Fakultas Teknologi Informasi, Program Studi Teknik Informatika, Universitas Merdeka Pasuruan, Jalan Ir Juanda No. 68, Pasuruan, Indonesia
[1] sidikrudini16@gmail.com *; [2] novidian@civitas.unas.ac.id ; [3] cianhassolthine@lecturer.unsia.ac.id ; [4] anang@unmerpas.ac.id

ARTICLE INFO

Article history
Received 2025/12/05
Revised 2025/12/15
Accepted 2025/12/20

Keywords
Recommendation System
Cosine Similarity
Term Frequency-Inverse Document Frequency
Rapid Application Development (RAD), Gemini

ABSTRACT

Problems with applications are potentially to disrupt business operational processes, especially in companies that depend entirely on applications. Therefore, speed and accuracy in handling every application problem that occurs is needed. One way to deal with various application problems effectively is to look for similar issues that have occurred before, and then take the handling solution as a reference for handling the current issue. This research aims to develop a recommendation system for handling application problems that can help the performance of the support services team. This system uses a cosine similarity algorithm with Term Frequency-Inverse Document Frequency weighting to find similar constraints based on the description. Before processing, the constraint description is summarized first using Gemini AI. Solutions to the obstacles found are used as a reference for handling current obstacles. The result of this research is that the system can summarize descriptions of issues and search for similar issues based on the dataset that has been trained. The recommendation system for handling application problems was well received by users, as evidenced by a score of 93.1% from 30 respondents who filled out the User Acceptance Test questionnaire.

1. Introduction

Software systems have become an inseparable part of the industrial sector in the modern era. Most companies rely on applications as primary tools to support their business operational processes. With high dependence on applications, the occurrence of issues or disruptions has the potential to hinder operational continuity. Therefore, rapid and accurate handling of every issue is crucial to ensure business processes remain efficient.

One effective approach to managing various application issues is leveraging experience from previously encountered similar issues. Solutions applied to past issues can serve as references for addressing current problems. In this context, content-based recommendation systems emerge as a relevant method.

Cosine similarity is an algorithm frequently used in content-based recommendation systems to measure the similarity between two vectors by calculating the angle between them. This method is commonly applied in unsupervised learning techniques [1]. A popular way to convert text into vectors is by using Term Frequency–Inverse Document Frequency (TF–IDF). For example, research by Lukman Heryawan et al. [2] utilized cosine similarity with TF–IDF weighting to build an effective medical document search model.

Previous studies have demonstrated the application of cosine similarity in various contexts. R.A. Purba [3] employed cosine similarity to detect plagiarism in undergraduate theses, where accuracy depended on word

count and spelling consistency. Testing on 30 documents showed a maximum similarity value of 41%. Indriyanto I. [4] also applied cosine similarity to detect plagiarism in Information Systems theses in Indonesia, demonstrating the method's effectiveness for Indonesian texts. Furthermore, Rika Rosnelly [5] used cosine similarity to assess the similarity of essay answers, incorporating stemming using the Nazief–Adriani algorithm prior to TF–IDF vectorization.

Based on this background, the present study implements cosine similarity with TF–IDF weighting to identify issues similar to currently occurring ones, using issue titles and descriptions as references. Prior to vectorization, titles and descriptions are summarized using Google's Gemini Large Language Model (LLM) to remove irrelevant text that could adversely affect similarity calculations.

In this study, cosine similarity and TF–IDF are applied to find similar issues based on the current issue's title and description. Before vectorization, titles and descriptions are summarized using Google Gemini LLM to remove irrelevant text and improve similarity calculation accuracy. This approach is expected to help support teams handle application issues more effectively and efficiently.

This research is limited to ticket report data from users recorded in a ticket management system, with a training dataset covering October 2024 to January 2025. The system is implemented as an additional feature in the existing application. The study aims to summarize issue descriptions to understand problems, identify similar issues by calculating the similarity between current and past issue descriptions, and use solutions from similar issues as references for support teams in handling new issues. This research is expected to benefit IT support teams in finding solutions more quickly, help companies improve efficiency in application issue handling, and provide a foundation for future research. The method applied uses cosine similarity to compute similarity on TF–IDF–weighted vectors of summarized issue descriptions, with summaries generated using Gemini AI to optimize similarity calculations.

2. Method

One of the important factors in information system development is how system developers understand the existing system and its associated problems. Therefore, a systematic data collection process using appropriate techniques is required to obtain a clear and comprehensive understanding of the system to be developed. The data collection techniques employed in this study are illustrated in the following figure.

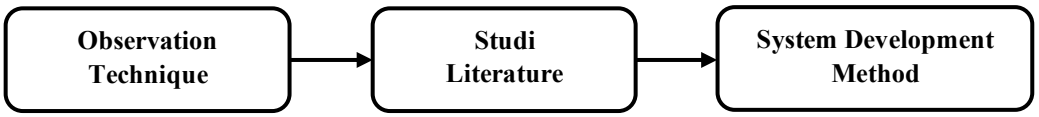

Fig. 1. Research Method

2.1 Observation Technique

Observation is one of the techniques commonly used in the data collection process to understand existing systems. According to Ramadhani, observation techniques can produce data with a high level of accuracy and reliability [6]. This technique is carried out by directly observing the research object, allowing the operational processes of the existing system to be clearly identified and understood. In this study, observations were conducted in the service management division, particularly within the support services department, which routinely handles issues arising from the managed applications. This data collection technique was carried out over a period of one month during the research process.

2.2 Studi Literature

At this stage, the authors conducted a literature review by searching for journals and articles related to the research object or the methods employed in the present study. This technique was used to understand previous studies and how related topics have been investigated by other researchers [7].

2.3 System Development Method

The system development method employed in this study is Rapid Application Development (RAD). RAD is a system development approach that emphasizes rapid development and short development cycles [7]. This method consists of several stages, which are illustrated in the following figure.

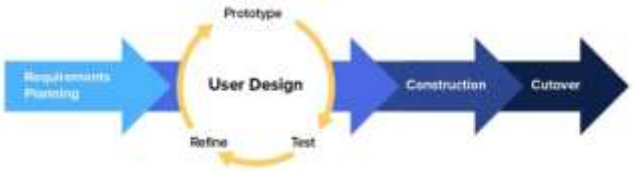

Fig. 2. Stages of the Rapid Application Development Method

2.3.1 Requirement Planning

The system development process begins with an analysis of initial requirements, which involves data collection and processing activities. The collected data are then analyzed to identify the requirements that serve as the foundation for the development process. This stage provides a comprehensive overview of the mechanisms required for developing an application issue-handling recommendation system [9].

2.3.2 User Design

The next stage involves designing the system to be developed. The outcome of this stage serves as a workflow reference that plays a crucial role in determining how the system will operate [9]. By producing a well-structured design, developers can implement a system that functions effectively and meets user requirements identified in the previous analysis stage [10]. This stage includes the design of the database schema, flowchart logic, and user interface layout.

2.3.3 Construction

The subsequent stage is the construction phase. At this stage, the system designed in the previous phase is implemented by developing program code in accordance with the defined requirements and models [8]. During this phase, the cosine similarity method with Term Frequency–Inverse Document Frequency (TF–IDF) weighting and support from Gemini AI is implemented to develop a recommendation system capable of identifying similar issues based on the currently reported issue.

2.3.4 Cutover

After the system has been fully developed, the next stage is testing, which aims to reduce the risk of errors within the system [8]. In this stage, User Acceptance Testing (UAT) is conducted to evaluate the extent to which the developed system meets user requirements and expectations [11], [20].

3. Results and Discussion

To achieve the results and discussion, this study undertakes several important processes, including :

3.1 Requirement Planning Stage

The first stage in developing the issue recommendation system is requirement planning. The authors planned the system requirements by conducting observations of the currently used system. Based on the observations, the proposed system will be developed as a REST API that can be integrated with the existing system. The system will be implemented using the Python programming language with the Flask framework. Python was chosen not only because of its widespread use in the field of machine learning but also due to its extensive library support for machine learning processes, such as Scikit-learn [12].

3.2 User Design Stage

At this stage, the authors begin designing the system to be implemented, including the following aspects:

- Database Design Stage
  Since this system will be implemented as an additional module within the existing system, the authors retained the use of the current database with only minor modifications.

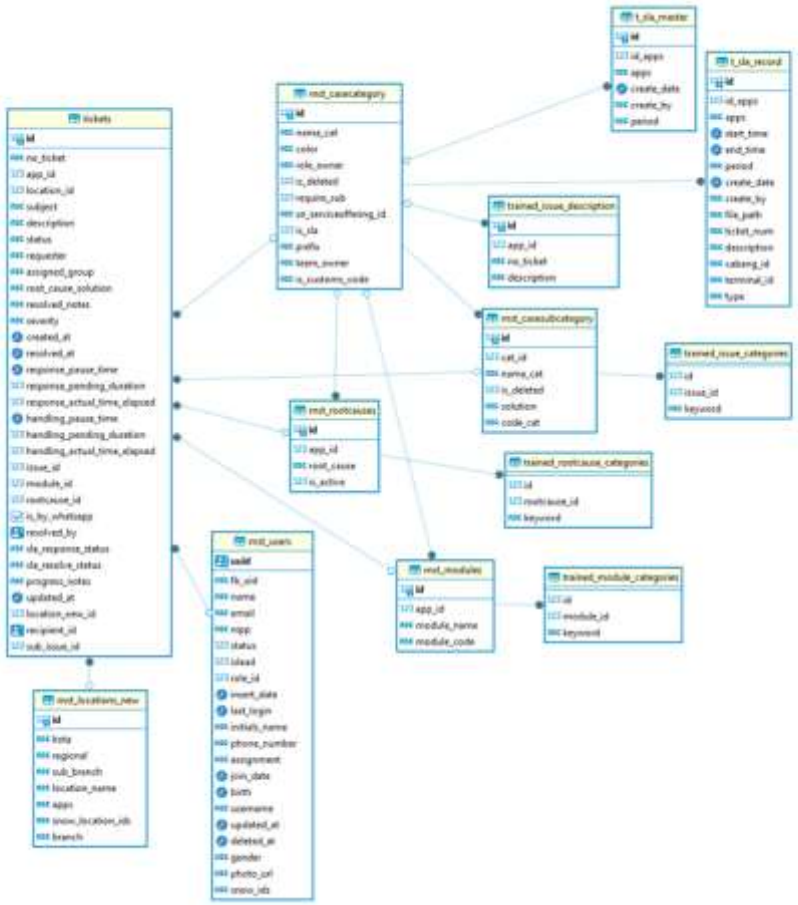

Fig. 3. Basis Data Design

The modification involved adding a new table to store training data, named trained_issue_description. The table structure is presented in Table 1 below.

Table 1. Structure of the trained_issue_description Table

No	Column Name	Data Tipe	Description
1	id	Serial	Primary Key
2	app_id	Int	Foreign Key
3	no_ticket	Varchar(255)	
4	description	Text	

As shown in Table 1, the trained_issue_description table consists of four columns. The app_id column is used to relate this table to the one containing the master data of applications. This linkage is useful for pre-filtering the training data to be used, ensuring that the process of identifying similar issues is focused and does not extend to unrelated applications.

- Flowchart Logic Design Stage
  Flowcharts are used as representations of system process flows, illustrating the sequence of activities and decisions within a program in a structured manner [13]. In the system design stage, this study implements two main flowcharts: the data training process flowchart and the similar issue search (similarity search) process flowchart. The data training process flowchart describes the mechanism for processing resolved issue data as training data. This data is periodically processed through a monthly training scheme to maintain model consistency and currency. The training stages include data selection, preprocessing, and storing the vectorized results in the database. The detailed training process flow is shown in Figure 4.
  The flowchart of the similar issue search process illustrates the system's computational workflow for generating relevant issue recommendations. The process begins with the collection of new issue data

from users, after which the system filters the training data based on application relevance. The selected training data then undergoes vectorization. The new issue is summarized using Gemini AI to produce a more concise and informative text representation, which is subsequently converted into a vector form. The system calculates similarity scores between the new issue vector and the training data vectors using a similarity measurement method. Based on these calculations, the system presents the three issues with the highest similarity scores as the output. The process flow is shown in Figure 5.

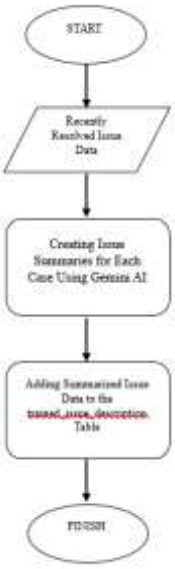

Fig.4. Data Training Process Flowchart

Fig. 5. Similar Issue Search Process Flowchart

- User Interface Design Stage
  At this stage, features were added to the system interface without altering the main structure of the existing system. The added feature includes a "Check Similarity" button in the issue detail menu (Figure 6). This button functions to trigger the similar issue search process, using the currently displayed issue as the search input. The system then processes the data and presents the search results in a modal window (Figure 7). The modal displays the title and summary of the current issue, as well as a list of other issues with similarity scores. During testing, the system identified only one similar issue.

Fig. 6. Check Similarity" on the Issue Detail Menu

Fig. 7. Modal Window for Similar Issue Search Results

3.3 Construction Stage

At this stage, the previously developed system design is implemented into program code, resulting in a functionally operable system. The construction process not only realizes the conceptual design into a tangible system but also allows for the evaluation of system performance, including functionality, process efficiency, and the alignment of outcomes with user requirements. The implementation results serve as the basis for analyzing the system's effectiveness and its impact on performance improvements compared to the previous condition.

The following provides a detailed explanation of the processes involved in the construction stage.

- Preparing the Gemini API Key

  In the implementation stage, the initial step involves preparing the Gemini Application Programming Interface (API) key as an authentication mechanism to access the artificial intelligence services provided by Google. The API key functions as an alphanumeric identifier that controls the application's access rights to the API services. An API itself is a communication mechanism that enables structured data exchange and integration between systems [14].

  The Gemini API key is obtained through the Google AI Studio platform by creating an access key under the developer account. Once the API key is generated, it is securely stored and kept confidential to prevent unauthorized access. In this study, the API key is not stored directly in the program code but is placed in an environment variable, following the recommended practice in previous research [15], to enhance system security and configuration flexibility.

Fig. 8. Google AI Studio Interface

Fig. 9. API Key Successfully Generated

- System Coding
  In the coding stage, the system was developed using the Python programming language with the Flask framework as the main structure for the web-based application. In addition to Flask, this study utilized several third-party libraries to support system functionality, including SQLAlchemy for database management and connectivity, scikit-learn for text weighting using TF–IDF and cosine similarity computation, google-generativeai for integration with the Gemini AI API, pandas for data processing and manipulation, and python-dotenv for managing environment variables in the development environment. The program code structure was organized modularly across several files to enhance readability and facilitate system maintenance. The file env_util.py is used to centralize environment variable retrieval, while gemini_util.py handles the prompting mechanism to Gemini AI during the issue summarization process. The primary database connection is managed through db_service.py using SQLAlchemy.

Fig. 10. Program Code in the env_util.py File

The database tables are represented as models, including MstApp, which represents the mst_casecategory table as the application master data, and TrainedIssueDescription, which represents the trained_issue_description table as storage for training data in the form of summarized resolved issues. At the controller layer, MstAppController handles requests for searching applications relevant to the reported issues, while SimilarityController manages the main process of identifying similar issues using TF–IDF and cosine similarity methods. Text vectorization is performed using TfidfVectorizer, and similarity scores are calculated using the cosine_similarity function from the scikit-learn library. The routing layer is implemented to connect client requests with the system logic, including the endpoints /get-app-ids and /check-similarity, which call their respective controllers. System configurations, including database connection settings, are defined in the config.py file, while server.py serves as the main entry point when the system is executed.

Fig. 11. Program Code in the env_util.py File

After all system components were fully implemented, the application was deployed to the server using a Docker container, enabling integration with and access by the interface module of the existing system.

- Adjustment of the Existing System Interface Module
  In the interface module of the existing system, minor modifications were made by adding a “Check Similarity” button and a modal to display the results of similar issue searches in the issue detail menu. These components were integrated with the services developed and deployed in the previous stage.

3.4 Cutover Stage

At this stage, the authors conducted User Acceptance Testing (UAT) by creating a questionnaire consisting of five questions. The questionnaire was then completed by selected personnel from the support team, who directly tested the newly developed features.

This stage can be described as follows:

- Summarizing Issue Descriptions

  One of the objectives in developing the issue-handling recommendation system was to summarize issue descriptions. This process is carried out before the system proceeds to the next stage, which is the similar issue search. Based on the development conducted, the recommendation system successfully achieved this objective by effectively summarizing issue descriptions, as illustrated in Figure 11 below.

Fig. 12. Issue Title and Description

The description was successfully condensed, as shown in Figure 13 below.

Fig. 13. Issue Title and Description

- Identifying Similar Issues

  The next expected outcome is that the system can identify similar issues to derive solutions as recommendations for handling the currently reported issue. This process represents the main feature implemented in the issue-handling recommendation system. Based on the conducted experiments, the developed system successfully identified similar issues corresponding to the current issue, as shown in Figure 13 above.

  In Figure 13, an example of a similar issue was identified: the inability to click on ordering. Based on cosine similarity calculations, this issue has a similarity score of 71% with the currently reported issue. The solution for the similar issue involved adjusting the in_dkk data in the ppkb table, and this solution was also applied to the current issue. This is illustrated in Figure 14 below, where the solution used to address the current issue closely matches the solution for the similar issue.

Fig. 14. Current Issue Resolution Results

- User Acceptance Testing
  To evaluate the extent to which the developed system meets user requirements, the authors conducted testing using the User Acceptance Testing (UAT) method [11]. A UAT questionnaire consisting of five items was created using a Likert scale. The Likert scale is a psychometric scale commonly used to measure the perceptions or attitudes of a group of individuals [16]. While the Likert scale is generally presented in five categories, other forms exist, ranging from simple "Yes" or "No" options to multiple-choice formats [17]. In this study, a three-point Likert scale was used, with values ranging from 1 to 3 [16], as shown in the following table.

Table 2. Questionnaire Assessment Categories

Answer	Value
Agree	3
Neutral	2
Disagree	1

In this testing, there were 30 respondents providing assessments, in accordance with the minimum number of respondents suggested in the study [18]. The results of the testing on the developed application issue-handling recommendation system are presented in Table 3 below.

Table 3. Questionnaire Results

Answer	Respondent Answers					
	Agree	Neutral	Disagree	Mean	Standard Deviation	Acceptance Percentage
Is the "Check Similar Issue" feature easy to use ?	26	4	0	2.87	0.34	86.7%
Does the "Check Similar Issue" feature help you search for information?	25	5	0	2.83	0.38	83.3%
Does the "Check Similar Issue" feature make your work easier?	22	7	1	2.73	0.48	73.3%
Are the results of the similar issue search consistent with the expected outcomes?	20	9	1	2.63	0.48	66.7%
Do you agree with using the "Check Similar Issue" feature?	28	2	0	2.93	0.26	93.3%

- Analysis

  Based on the implementation results of the developed system, the research objectives were successfully achieved, including summarizing issue descriptions and identifying similar issues to assist the support team in handling reported problems. According to the conducted experiments, the system was able to summarize issue descriptions and identify similar issues effectively. The authors also compared the handling of the identified similar issues with the solutions applied to the issue used as a test case, and the recommended solutions closely matched those implemented for the actual issue. Furthermore, the authors conducted a detailed analysis of user acceptance based on the questionnaire results presented in Table 4.2 above. The calculation of acceptance percentages was performed using the Likert scale method [19] as follows:

  ✓ $\text{Acceptance Percentage} = \frac{\text{total Likert score}}{\text{maximum Likert score}} \times 100\%$

  ✓ $\text{Acceptance Percentage} = \frac{(121\times3)+(27\times2)+(2\times1)}{450}$

  ✓ Acceptance Percentage = 93.1%

  The analysis results indicate that user acceptance of the developed application issue-handling recommendation system is very high, with a score of 93.1%. The majority of users agreed with each item in the questionnaire, demonstrating that the developed system effectively assists users in finding similar issues, thereby facilitating their work.

4. Conclusion

Based on the implementation and testing results, it can be concluded that the developed application issue-handling recommendation system is capable of supporting the speed and accuracy of issue resolution in operational environments dependent on applications. The system utilizes the cosine similarity algorithm with TF–IDF weighting to identify similar issues based on descriptions summarized using Gemini AI, allowing solutions from previous issues to serve as references for current issue handling. Testing results demonstrate that the system can effectively perform issue description summarization and similar issue search based on the trained dataset. Furthermore, user acceptance of the system is very high, as evidenced by a User Acceptance Test score of 93.1% from 30 respondents, indicating that the system is suitable for implementation to enhance the performance of the support team.

References

[1] Aristoteles, M. U. Syam, Tristiyanto, dan B. Hermanto, “Implementation of Cosine Similarity Algorithm on Omnibus Law Drafting,” International Journal of Advanced Computer Science and Applications, vol. 15, no. 4, hlm. 200–205, 2024, [Daring]. Tersedia pada: www.ijacsa.thesai.org

[2] L. Heryawan, D. Novitaningrum, K. R. Nastiti, dan S. N. Mahmudah, “Medical Record Document Search with TF-IDF and Vector Space Model (VSM),” International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, 2024.

[3] R. A. Purba, S. Suparno, dan M. Giatman, “The optimalization of cosine similarity method in detecting similarity degree of final project by the college students,” dalam IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Mei 2020, hlm. 1–6. doi: 10.1088/1757-899X/830/3/032003.

[4] I. Indriyanto dan I. D. Sumitra, “Measuring the Level of Plagiarism of Thesis using Vector Space Model and Cosine Similarity Methods,” dalam IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Nov 2019, hlm. 1–6. doi: 10.1088/1757-899X/662/2/022111. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] R. Rosnelly, D. Hartama, M. Sadikin, C. P. Lubis, M. S. Simanjuntak, dan S. Kosasi, “The Similarity of Essay Examination Results using Preprocessing Text Mining with Cosine Similarity and Nazief-Adriani Algorithms,” Turkish Journal of Computer and Mathematics Education, vol. 12, no. 3, hlm. 1415–1422, 2021. R. Rosnelly, D. Hartama, M. Sadikin, C. P. Lubis, M. S. Simanjuntak, dan S. Kosasi, “The Similarity of Essay Examination Results using Preprocessing Text Mining with Cosine Similarity and Nazief-Adriani Algorithms,” Turkish Journal of Computer and Mathematics Education, vol. 12, no. 3, hlm. 1415–1422, 2021.

[6] M. S. Ramadani dan A. Munazilin, "Sistem Informasi Arsip Surat Berbasis Web Pada Dinas Perikanan Kabupaten Banyuwangi," Gudang Jurnal Multidisiplin Ilmu, vol. 2, no. 10, hlm. 333–339, 2024, doi: 10.59435/gjmi.v2i10.1008.

[7] L. Nilawati, D. Sulastri, dan Y. Yuningsih, "Penerapan Model Rapid Application Development Pada Perancangan Sistem Informasi Jasa Pengiriman Barang," Paradigma - Jurnal Informatika dan Komputer, vol. 22, no. 2, hlm. 197–204, 2020, doi: 10.31294/p.v21i2.

[8] N. Hidayat dan K. Hati, "Penerapan Metode Rapid Application Development (RAD) dalam Rancang Bangun Sistem Informasi Rapor Online (SIRALINE)," Jurnal Sistem Informasi STMIK Antar Bangsa, vol. X, no. 1, hlm. 8–17, 2021.

[9] Y. D. Wijaya, "Penerapan Metode Rapid Application Development (Rad) Dalam Pengembangan Sistem Informasi Data Toko," Jurnal Sistem Informasi dan Teknologi, vol. 3, no. 2, hlm. 95–102, 2020, [Daring]. Tersedia pada: http://www.jurnal.umk.ac.id/sitech

[10] Supriyanta, E. Rahmawati, dan I. H. Basri, "Perancangan Sistem Informasi Pengelolaan Arsip Berbasis Web Dengan Metode Prototype," Indonesian Journal on Software Engineering (IJSE), vol. 10, no. 1, hlm. 52–62, 2024, [Daring]. Tersedia pada: http://ejournal.bsi.ac.id/ejurnal/index.php/ijse52

[11] R. R. Hidayat, M. Fikry, dan Yusra, "Chatbot Deteksi Awal Gangguan Kecemasan Menggunakan Dialogflow," Jurnal Teknologi Terpadu, vol. 11, no. 2, hlm. 293–302, 2023.

[12] M. R. sirfatullah Alfarizi, M. Z. Al-farish, M. Taufiqurrahman, G. Ardiansah, dan M. Elgar, "Penggunaan Python Sebagai Bahasa Pemrograman Untuk Machine Learning Dan Deep Learning," Karimah Tauhid, vol. 2, no. 1, hlm. 1–6, 2023.

[13] A. Zalukhu, S. Purba, dan D. Darma, "Perangkat Lunak Aplikasi Pembelajaran Flowchart," Jurnal Teknologi Informasi dan Industri, vol. 4, no. 1, hlm. 61–70, 2023.

[14] N. Rachmat dan D. P. Kesuma, "Implementasi Large Language Models Gemini Pada Pengembangan Aplikasi Chatbot Berbasis Android," Jurnal Ilmu Komputer (JUIK), vol. 4, no. 1, hlm. 40–52, 2024, [Daring]. Tersedia pada: https://journal.umgo.ac.id/index.php/juik/index

[15] A. R. R. Putri, I. Arwani, dan W. Purnomo, "Pengembangan Sistem Informasi Monitoring dan Pembayaran Jasa Perbaikan Komputer (Studi Kasus: CV Mitra Solusi Gresik)," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 6, no. 3, hlm. 1058–1066, 2022, [Daring]. Tersedia pada: http://j-ptiik.ub.ac.id`

[16] A. Aditya Santika, T. Hamonangan Saragih, Muliadi, D. Kartini, dan R. Ramadhani, "Penerapan Skala Likert Pada Klasifikasi Tingkat Kepuasan Pelanggan Agen BRILink Menggunakan Random Forest," Jurnal Sistem dan Teknologi Informasi, vol. 11, no. 3, hlm. 405–411, 2023, doi: 10.26418/justin.v11i3.

[17] A. H. Suasapha, "Skala Likert Untuk Penelitian Pariwisata; Beberapa Catatan Untuk Menyusunnya Dengan Baik," Jurnal Kepariwisataan, vol. 19, no. 1, hlm. 26–37, Mar 2020, doi: 10.52352/jpar.v19i1.407.

[18] K. Nurazizah dan I. Mildawani, "Persepsi Dan Preferensi Masyarakat Terhadap Implementasi Citra Arsitektur Pecinan Di Jalan Kisamaun Tangerang," UG Journal, vol. 16, no. 8, hlm. 27–37, 2022.

[19] S. Purwanti dan R. Z. A. Putri, "Pengembangan Modul Berbasis Hots Pada Tema 6 Materi Membandingkan Siklus Makhluk Hidup Kelas Iv Sekolah Dasar," Elementary School, vol. 8, no. 1, hlm. 155–160, 2021.

[20] Widodo, A. A., Almais, A. T. W., Alamsyah, M., & Hariyanto, R. (2022). *Usability analysis to measure the effectiveness of implementing the mapping system for COVID-19 patients*. In Proceedings of the 2nd International Conference on Information Technology and Education (ICIT&E 2022) (pp. 92–96). IEEE. https://doi.org/10.1109/ICITE54466.2022.9759838