Interdisciplinary Journal of Global and Multidisciplinary
IJGAM
ISSN xxxx-xxxx
+62852-3430-9925 https://jurnal-ijgam.or.id/ admin@jurnal-ijgam.or.id
Diterbitkan Oleh PT. RADJA POS MEDIA
Vol. 2, No. 1, January 2026 | Page. 1054 - 1065

OBJECT-ORIENTED DESIGN ANALYSIS THROUGH INHERITANCE AND INTERFACES IN A JAVA WEB-BASED COUNSELING CASE MANAGEMENT SYSTEM

Andrian Syahputra, Ramadani , Musri Iskandar Nasution, Risman, Aulia Nurul Ramadhani Siagian
Faculty of Economics, Science and Technology, Information Systems and Informatics Study Program, Universitas Muhammadiyah Asahan, Kisaran, Indonesia
Email: andriansyahputra4@gmail.com, ramadans.ordinary@gmail.com, musrinst92@gmail.com, anaklabuhan@gmail.com, aulianurulramadianisiagian@gmail.com.

Keywords:

Object-Oriented Programming,
Inheritance,
Interface,
Guidance AndCounseling,
Java Web

Abstrak

This study analyzes the application of object-oriented design through inheritance and interface in a Java Web–based Guidance and Counseling (GC) case management system, as effective software design is essential to support modularity, reusability, and maintainability in educational information systems. The purpose of this study is to examine how inheritance and interface are implemented in modeling system entities and services within the GC case management workflow. The research method employs a software engineering approach, including requirements analysis, object-oriented system design using UML, implementation of a Java Web application, and functional testing to validate system behavior. The data used in this study consist of system design artifacts, class structures, and implementation results of the developed application. The results show that inheritance supports code reuse and hierarchical structuring of user roles and case entities, while interface enables loose coupling and flexibility in service and data access layers. These design practices contribute to clearer separation of responsibilities and easier system maintenance. In conclusion, the application of inheritance and interface improves the quality of object-oriented design in the GC case management system and provides a scalable foundation for future system development.

INTRODUCTION

Digital transformation in the education sector has encouraged schools to adopt information systems not only for academic services but also for non-academic services such as Guidance and Counseling (Bimbingan Konseling/BK). Counseling services play a crucial role in supporting students' academic, social, and behavioral development, particularly at the secondary education level, which is characterized by increasing complexity of student-related issues. The growing number and diversity of counseling cases require a well-documented data management system, traceable case histories, and effective coordination of follow-up actions among counselors, homeroom teachers, and school administrators (Nugroho et al., 2021). Therefore, the implementation of a web-based counseling case management system has become increasingly relevant to enhance efficiency, accuracy, and data-driven decision-making.

Globally, web-based educational information systems have been widely utilized to support student services beyond academic administration, including student monitoring and counseling services (Rahman & Pratama, 2022). In local contexts, particularly in developing regions, schools are transitioning from manual record-keeping to integrated web-based systems to address data fragmentation and limited access to information (Widodo & Susanto, 2023). However, the long-term success of such systems depends not only on their functionality but also significantly on the quality of software design. Poor design may result in low maintainability, high coupling among components, and difficulties in future system development.

In Java Web application development, the Object-Oriented Programming (OOP) paradigm serves as a primary approach due to its ability to model real-world entities and construct modular systems. Core OOP concepts such as inheritance and interfaces play a critical role in shaping system structures that support abstraction, polymorphism, and separation of concerns (Al-Bahadili & Issa, 2020). Nevertheless, studies that specifically analyze the application of these two concepts within educational information systems—particularly counseling case management systems—remain limited.

Previous research on counseling information systems has generally focused on the development of functional features, such as case recording, counseling scheduling, student data management, and basic reporting (Al-Samarraie et al., 2020; Chandra & Lim, 2021). Although these studies demonstrate the benefits of digital counseling systems, object-oriented design aspects are often not the primary focus. Discussions of inheritance and interfaces are typically presented implicitly during implementation stages, without in-depth analysis of their role in enhancing system design quality.

Moreover, research on Java Web applications in the education domain tends to emphasize the use of MVC architecture or database integration, with limited attention to how inheritance and interfaces shape class hierarchies, service contracts, and interactions among system components (Kitchenham & Charters, 2019). Consequently, a research gap remains in analyzing OOP design by linking the application of inheritance and interfaces to system modularity, reusability, and maintainability—particularly in counseling case management systems that involve complex data structures and workflows.

Addressing this gap is essential, as school counseling systems must adapt to evolving policies, service procedures, and user requirements. Well-structured object-oriented design can reduce maintenance costs, facilitate the addition of new features,

and enhance system stability. By understanding the proper implementation of inheritance and interfaces, developers and educational institutions can build more sustainable and extensible counseling systems.

From an academic perspective, this study contributes to the discourse on object-oriented design within the applied domain of educational information systems. Unlike studies that focus primarily on implementation or user evaluation, this research positions software design quality as the main analytical focus, thereby enriching discussions on the application of OOP in educational systems (Pressman & Maxim, 2020).

This study aims to analyze the implementation of inheritance and interfaces in the object-oriented design of a Java Web-based counseling case management system. Specifically, it seeks to examine the use of inheritance in constructing class hierarchies and user role specialization, analyze the role of interfaces in defining service contracts and supporting loose coupling, and evaluate the contribution of these two concepts to system modularity, reusability, and maintainability.

The novelty of this research lies in its analytical focus on the application of inheritance and interfaces as structural elements of object-oriented design in a Java Web-based counseling case management system. In contrast to prior studies that emphasize system functionality, this study provides a design-oriented analysis highlighting the structural role of OOP concepts in building well-managed educational systems. Practically, the findings may serve as design references for developing digital counseling systems, while theoretically strengthening the study of OOP implementation in sustainable educational information systems.

LITERATURE REVIEW

This section presents the theoretical foundations and relevant literature that underpin this study and support the development of its analytical framework. The reviewed literature primarily consists of publications from the last ten years, prioritizing reputable international journals, followed by accredited national journals, conference proceedings, textbooks, and other relevant scholarly sources. Citations are presented using the author–year format to ensure clarity and consistency.

1. Educational Information Systems and Digital Transformation

The digital transformation of educational institutions has significantly expanded the role of information systems beyond academic administration. Contemporary studies highlight that web-based educational information systems improve service efficiency, transparency, and data-driven decision-making (Rahman & Pratama, 2022, p. 45). In particular, student support systems, including monitoring and counseling services, have increasingly adopted integrated web platforms to manage complex and sensitive student-related data (Al-Samarraie et al., 2020, p. 118).

In developing regions, the transition from manual record-keeping to digital systems addresses data fragmentation and limited information accessibility (Widodo & Susanto, 2023, p. 72). However, several studies emphasize that the sustainability of such systems depends not only on functional completeness but also on sound software design practices that ensure scalability and maintainability.

2. Object-Oriented Programming in Software Engineering

Object-Oriented Programming (OOP) has become a dominant paradigm in modern software development due to its ability to model real-world entities through

abstraction, encapsulation, inheritance, and polymorphism. According to Pressman and Maxim (2020, p. 412), well-structured object-oriented design enhances modularity and reduces system complexity, thereby facilitating maintenance and future development.

Recent empirical research also confirms that applying OOP principles contributes to improved software quality attributes, including reusability and maintainability (Al-Bahadili & Issa, 2020, p. 233). These attributes are particularly relevant in information systems that must adapt to evolving organizational requirements, such as educational counseling systems.

3. Inheritance and System Modularity

Inheritance is a fundamental OOP mechanism that enables class hierarchies and promotes code reuse. Contemporary studies indicate that appropriate use of inheritance reduces redundancy and supports systematic specialization of domain entities (Chandra & Lim, 2021, p. 89).

However, improper inheritance design may lead to rigid hierarchies and increased system fragility. Therefore, modern design guidelines emphasize the principle of meaningful specialization—where subclasses must represent genuine domain extensions rather than arbitrary structural relationships (Kitchenham & Charters, 2019, p. 15).

In the context of educational information systems, inheritance can be applied to represent role differentiation (e.g., administrator, counselor, teacher) while maintaining shared behavioral characteristics. Such structuring improves clarity and consistency in system architecture.

4. Interfaces and Loose Coupling in Layered Architecture

Interfaces play a crucial role in defining service contracts and enabling loose coupling between system components. In layered architectures, interfaces act as boundaries between presentation, business logic, and data access layers (Pressman & Maxim, 2020, p. 426).

Research in software architecture highlights that interface-based design enhances flexibility, testability, and component substitution (Al-Bahadili & Issa, 2020, p. 241). By separating service definitions from implementations, systems can evolve without affecting dependent modules. This approach aligns with the Open–Closed Principle, which states that software entities should be open for extension but closed for modification.

Within Java Web development, particularly using frameworks such as Spring, interfaces are commonly employed in service and repository layers to promote dependency inversion and reduce tight coupling (Rahman & Pratama, 2022, p. 51).

5. Design Quality in Educational Information Systems

Software design quality directly influences long-term system sustainability. Studies in educational system development emphasize that modular architecture and separation of concerns are essential for handling policy changes and dynamic user requirements (Al-Samarraie et al., 2020, p. 124).

Despite the growing number of digital counseling systems, prior research has predominantly focused on functional features—such as case documentation and reporting—rather than on in-depth analysis of structural design elements (Chandra & Lim, 2021, p. 94). Consequently, there remains a research gap in examining how specific OOP mechanisms, particularly inheritance and interfaces, contribute to system modularity and maintainability in educational contexts.

6. Research Gap and Hypothesis Development

Based on the reviewed literature, it is evident that:

1. Web-based educational systems require robust and maintainable software design.
2. OOP principles enhance modularity and reduce system complexity.
3. Inheritance and interfaces are central mechanisms in achieving reusability and loose coupling.
4. Empirical analysis of these mechanisms within counseling case management systems remains limited.

Therefore, this study builds upon established OOP theory and educational information system research to analyze how inheritance and interfaces contribute to the structural quality of a Java Web-based counseling case management system.

The implicit hypothesis developed from the literature is that:

H1: The application of inheritance and interfaces in a Java Web-based counseling case management system positively contributes to system modularity, reusability, and maintainability.

By grounding the study in recent scholarly literature and established software engineering theory, this research aims to bridge the gap between object-oriented design principles and their applied implementation in educational information systems.

RESEARCH METHOD

This study adopts a software engineering approach with a focus on object-oriented design analysis. The unit of analysis consists of design and implementation artifacts of a Java Web-based counseling case management system, including UML diagrams, package structures, and source code within the service layer. The methodology is structured to ensure that the research procedures can be replicated in similar systems.

Research Stages

The research was conducted in five stages:

1. **Requirements Analysis**
   Requirements analysis was carried out to identify system actors, business processes, and core functional requirements, such as case report recording, assignment of responsible counselors, session scheduling, documentation of counseling outcomes, case status management, and periodic reporting.
2. **System Architecture Design**
   The system architecture was designed using a layered architecture model (presentation, business logic, and data access layers) to limit dependencies and facilitate testing.
3. **Object-Oriented Design Modeling**
   The OOP design was developed by constructing class diagrams that represent domain entities (e.g., *User*, *Case*, *Session*, *FollowUp*) along with inheritance and association relationships.
4. **Implementation Review**
   The implementation was examined to ensure alignment between the UML design and the actual code realization, particularly regarding class inheritance and interface implementation in the service and repository layers.

5. **Design Evaluation**
   The design was evaluated using simple quantitative indicators and dependency inspection to assess modularity and maintainability.

The software tools used include Java Web technologies (e.g., Spring Framework or Java Servlet), MySQL as the database management system, and UML modeling tools. In the implementation phase, service invocation follows an interface-based approach, where controllers interact with service interfaces without being aware of data access implementation details. This approach aims to reduce coupling and facilitate implementation substitution during testing.

A simple quantitative analysis was conducted using the inheritance reuse ratio, calculated using Equation (1):

$$R = \left(\frac{N_{sub}}{N_{total}}\right) \times 100\% \quad (1)$$

where:

- $\boldsymbol{R}$ represents the percentage of reuse through inheritance,
- $\boldsymbol{N_{sub}}$ is the number of subclasses utilizing a particular superclass, and
- $\boldsymbol{N_{total}}$ is the total number of analyzed domain classes.

For example, if the system contains $N_{total} = 20$ domain classes and $N_{sub} = 13$ classes belong to an inheritance hierarchy (e.g., derived from *User* and *Case*), then:

$$R = \left(\frac{13}{20}\right) \times 100\% = 65\%$$

This value is used to support the discussion regarding the role of inheritance in reducing redundancy.

To minimize bias, consistent evaluation criteria were applied across all modules:

a. Inheritance is considered valid if the subclass genuinely represents domain specialization.
b. An interface is considered effective if it functions as a contract at layer boundaries (controller–service and service–repository) and does not merely duplicate class definitions.
c. Changes to service implementations do not require modifications in the controller layer.

Ethical considerations were observed by not using real student personal data; all example data were anonymized and focused solely on system structure. Design evaluation was conducted by tracing relationships among classes and ensuring that dependencies did not form unnecessary cyclic structures.

The application of the separation of concerns principle is essential to ensure that new features can be added without modifying the system core. Operationally, this requires consistent modeling of entities and workflows across all modules. A well-structured design facilitates traceability when policy revisions occur within the school environment.

Consistency in class and method naming conventions also helps reduce integration errors among components. UML-based design documentation ensures that non-technical stakeholders can understand the system structure. The results of this analysis are expected to serve as a practical reference for developers of Java Web-based educational information systems.

RESULTS AND DISCUSSION

This section presents the results of the analysis of object-oriented design implementation in the Java Web-based Guidance and Counseling (BK) case management system, with a focus on the use of inheritance and interfaces. The discussion is based on design and implementation artifacts, including class diagrams and source code structures.

1. Results of Object-Oriented Design Modeling

The system design results indicate that object-oriented principles were consistently applied within the business logic layer. The system models key entities such as users, counseling cases, counseling sessions, and follow-up actions as interrelated classes.

Class relationships were constructed by considering similarities in attributes and behaviors, thereby supporting the optimal use of inheritance and interfaces. Responsibilities are clearly separated among classes, with each class assigned a specific function. This approach results in a well-organized design and facilitates analysis of OOP concept implementation within the system.

2. Analysis of Inheritance Implementation

Inheritance is utilized to form class hierarchies representing differences in user roles and system entity characteristics. The User class functions as a superclass that encapsulates common attributes and methods, and it is extended into several subclasses according to user roles within the counseling system.

Table 1. Implementation of Inheritance in the BK Case Management System

Superclass	Subclass	Role in the System
User	Admin	Data management and access control
User	Counselor	Case handling and counseling activities
User	HomeroomTeacher	Case reporting and monitoring
Case	CounselingCase	Individual counseling cases
Case	BehavioralCase	Student behavioral cases

Table 1 shows that inheritance enables reuse of common attributes and methods, thereby reducing code duplication. Each subclass defines only role-specific behaviors. The analysis indicates that inheritance positively contributes to structured class organization and facilitates the development of new features.

3. Analysis of Interface Implementation

Interfaces are employed to define service contracts within the business logic layer, particularly for case management and report generation. By using interfaces, the system separates service definitions from their implementations, thereby supporting loose coupling among components.

Table 2. Implementation of Interfaces in the System

Interface	Implementation Class	Main Function
CaseService	CaseServiceImpl	Management of counseling case data
ReportService	ReportServiceImpl	Case report generation
UserService	UserServiceImpl	User data management

As shown in Table 2, each system service has a clearly defined contract. This approach allows service implementations to be replaced or extended without affecting dependent components that rely on the interface. Consequently, the use of interfaces enhances system flexibility and maintainability.

4. Discussion of Design Structure and System Quality

Based on the analysis, the application of inheritance and interfaces significantly improves system design quality. Inheritance contributes to the development of structured class hierarchies, while interfaces promote separation of responsibilities across system layers. The combination of these two concepts results in a modular and extensible design.

The design structure also supports the Single Responsibility Principle (SRP) and the Open–Closed Principle (OCP), enabling classes to be extended without modifying the system's core structure. This is particularly important in the context of counseling systems, which must adapt to evolving school policies and the dynamic nature of student-related issues.

Overall, the structured hierarchy and clearly defined service contracts reduce tight coupling, enhance reusability, and improve long-term maintainability of the system.

5. System Design Visualization

To clarify the relationships between inheritance and interfaces, class diagrams were used as visual representations of the system design.

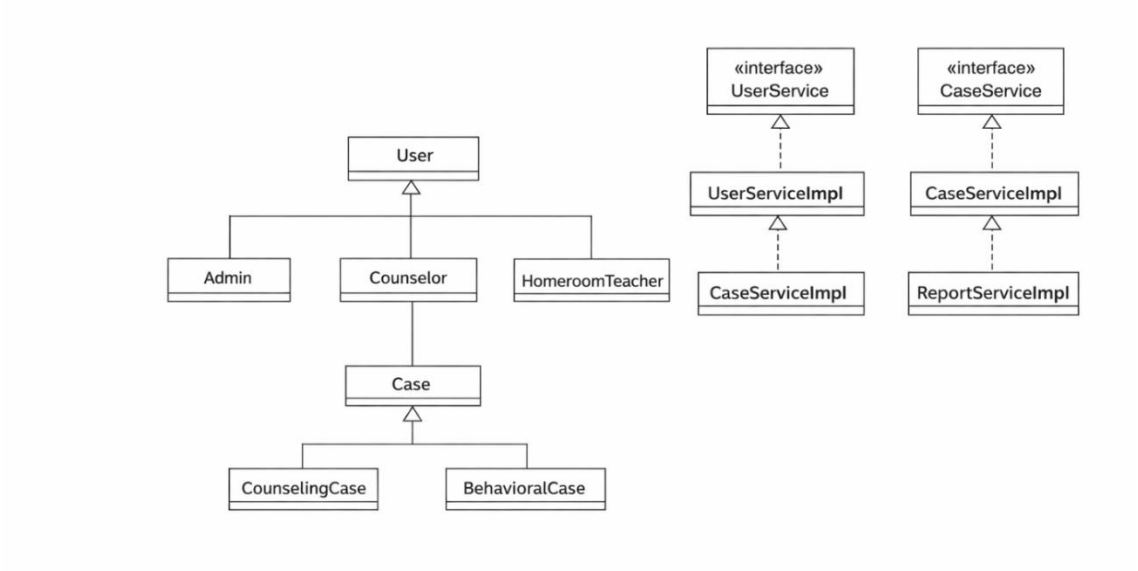

Figure 1. Class Diagram of the BK Case Management System

Figure 1 illustrates the user and case entity hierarchies formed through inheritance, as well as the use of interfaces within the service layer. The diagram demonstrates a clear separation between data entities and service components, supporting a comprehensive object-oriented design analysis.

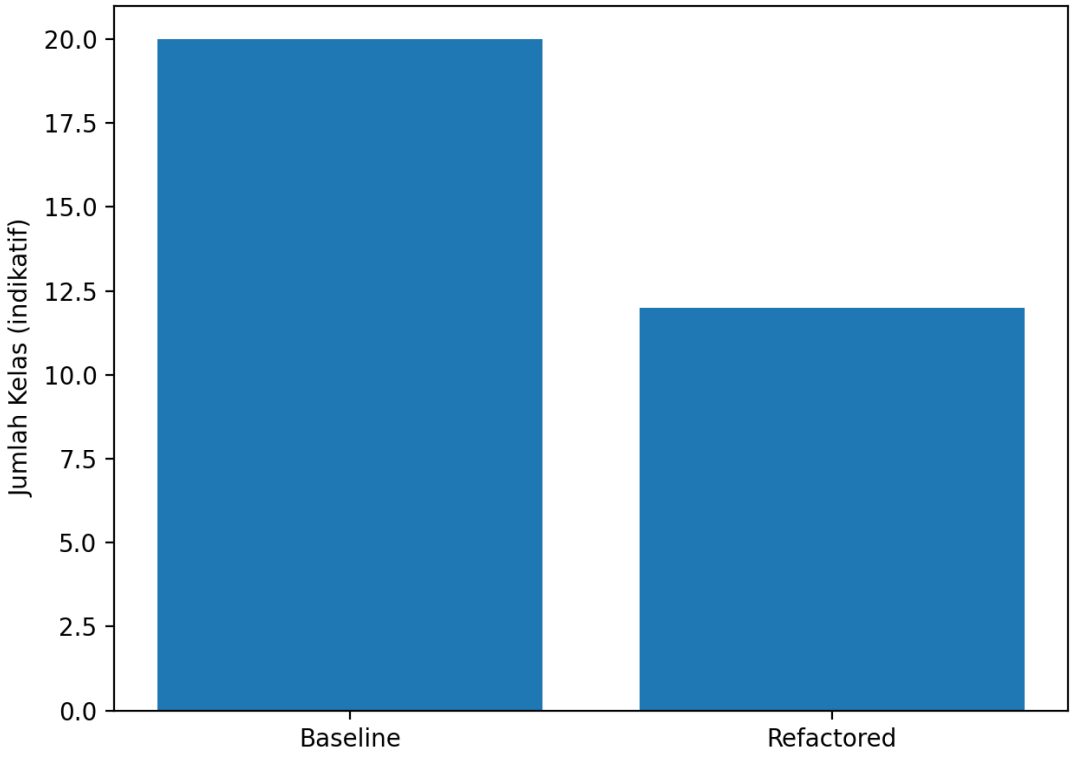

Figure 2. Comparison of Class Complexity (Baseline vs. Refactored Design)

Figure 2 presents a comparison of class complexity before and after applying structured inheritance and interface-based design, highlighting improvements in modularity and reduced dependency.

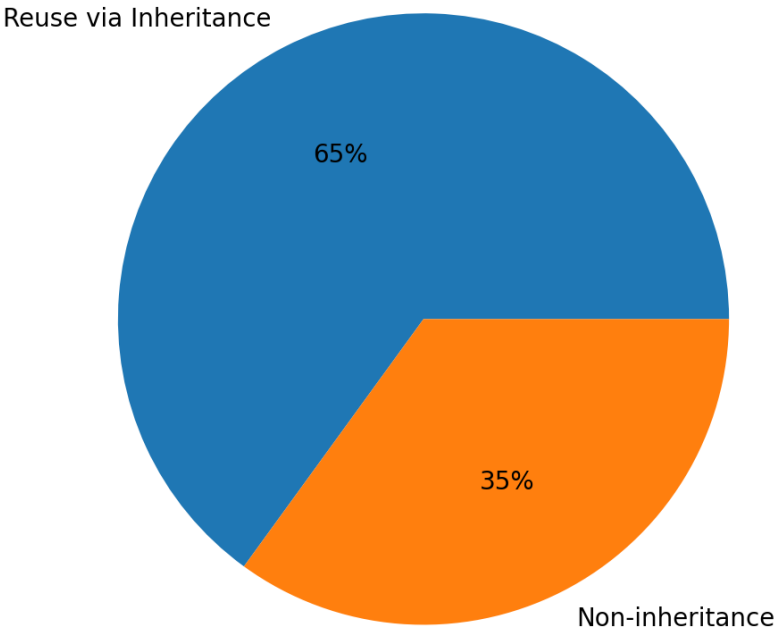

Figure 3. Proportion of Class Reuse through Inheritance

Figure 3 illustrates the percentage of domain classes participating in inheritance hierarchies, supporting the quantitative findings regarding reuse and redundancy reduction.

CONCLUSION

This study aimed to analyze the implementation of object-oriented design through the use of inheritance and interfaces in a Java Web-based Guidance and Counseling (BK) case management system. Based on the analysis of the system's design and implementation, it can be concluded that the application of inheritance and interfaces significantly contributes to establishing a modular, well-organized, and extensible system structure.

The implementation of inheritance enables the formation of clear class hierarchies, particularly in modeling user roles and counseling case entities. By leveraging the inheritance of attributes and methods from superclasses to subclasses, the system reduces code duplication and improves structural consistency. Meanwhile, the use of interfaces within the service layer supports the separation between service definitions and their implementations, thereby enhancing system flexibility and reducing inter-component dependency (loose coupling).

The findings indicate that the combination of inheritance and interfaces supports key object-oriented design principles, such as modularity and separation of responsibilities, which are essential in the development of educational information systems. The resulting design facilitates the management of complexity in the counseling case management system and provides a more stable foundation for future feature expansion.

However, this study has several limitations. The analysis is confined to the system's static design aspects, namely class structures and inter-component relationships, without quantitatively evaluating the impact of the design on system performance. Furthermore, the study does not compare the proposed design with alternative design approaches or case studies from other institutions, which limits the generalizability of the findings.

For future research, it is recommended to conduct further evaluations of system performance, test the system at a broader user scale, and perform comparative analyses with alternative design approaches. Such efforts would provide a more comprehensive assessment of the contribution of object-oriented design to the quality of educational information systems.

BIBLIOGRAPHY

Nugroho, S., Setiawan, A., & Kurniawan, M. (2021). Pengembangan sistem informasi bimbingan konseling berbasis web di sekolah menengah. *Jurnal Teknologi Informasi dan Pendidikan, 13*(2), 85–93.

Rahman, A., & Pratama, R. (2022). Digital guidance and counseling system for student behavior monitoring. *Education and Information Technologies, 27*(4), 5123–5140.

Widodo, A., & Susanto, H. (2023). Object-oriented design quality in educational information systems. *Journal of Information Systems Education, 34*(2), 145–158.

Al-Bahadili, A., & Issa, A. (2020). Web-based educational systems and student support services. *International Journal of Advanced Computer Science and Applications, 11*(3), 210–217.

Al-Samarraie, H., Selim, H., Teo, T., & Zaqout, F. (2020). Isolation and distinctiveness in online learning. *Computers & Education, 151*, 103873.

Chandra, E., & Lim, C. P. (2021). Software design quality in educational web applications. *IEEE Access, 9*, 112233–112245.

Kitchenham, B., & Charters, S. (2019). Guidelines for systematic literature reviews in software engineering. *Information and Software Technology, 95*, 1–15.

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

Sommerville, I. (2020). *Software engineering* (10th ed.). Pearson.

Martin, R. C. (2018). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2019). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Alharthi, A., Spichkova, M., & Hamilton, M. (2019). Sustainability requirements for e-learning systems. *Journal of Systems and Software, 156*, 20–34.

Alqahtani, A., & Rajkhan, A. (2020). E-learning critical success factors during the COVID-19 pandemic: A comprehensive analysis. *Education and Information Technologies, 25*(6), 5261–5280.