CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Design and Development of a Web-Based Virtual Programming Laboratory for Information Systems Education

David Naista[1*], Ghifar Javad H Aziz[2]
[1,2]Information Systems, UIN Raden Intan Lampung, Indonesia
[1*]davidnaista@radenintan.ac.id, [2]ghifarjavad@radenintan.ac.id

Abstract: This study presents the design, development, and evaluation of a web-based virtual programming laboratory for Information Systems education. The research addresses key limitations of traditional programming laboratories, including limited accessibility, dependency on physical infrastructure, and restricted opportunities for continuous practice. To overcome these challenges, a virtual laboratory system is developed using a web-based architecture that enables users to perform programming activities anytime and anywhere. The system adopts a layered architecture consisting of presentation, application, execution, and data layers. Core features include a web-based code editor, a server-side execution engine, structured learning modules, and real-time output visualization. The system is implemented as an integrated platform that supports interactive programming practice and instructional content delivery. The evaluation is conducted through functional testing, usability assessment, and performance analysis. The results indicate that the system operates reliably, with all core functionalities performing as expected. Usability evaluation shows that the system provides a clear interface, efficient navigation, and high accessibility across devices. The performance analysis demonstrates that the system supports real-time interaction with acceptable response time. From a pedagogical perspective, the system enhances learning effectiveness by enabling continuous practice and providing immediate feedback. The structured learning modules support progressive skill development and improve problem-solving ability. The findings confirm that the developed virtual laboratory provides an effective and scalable solution for programming education. The system contributes to advancing digital learning environments and offers practical implications for higher education institutions.
Keywords: virtual laboratory; web-based learning; programming education; information systems; e-learning;

1. INTRODUCING

The rapid advancement of information and communication technology has fundamentally transformed the structure and delivery of higher education. Digital learning environments have become an integral component in modern academic systems, particularly in disciplines that require both theoretical understanding and practical skills. In Information Systems education, programming courses represent a core element that demands continuous practice to develop computational thinking, logical reasoning, and problem-solving capabilities. [1], [2]

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Programming is not solely a theoretical discipline. It requires active engagement through repeated coding activities, experimentation, and debugging. Students must translate abstract concepts into executable instructions, which involves a complex cognitive process. As a result, effective programming education depends heavily on the availability of practical learning environments that support continuous and iterative practice.

Traditional programming laboratories have been widely used to facilitate practical learning. However, these environments present several structural limitations. First, access to laboratory facilities is typically restricted to scheduled sessions, which limits the time available for students to practice. This restriction prevents students from exploring programming concepts beyond classroom hours. Second, laboratory infrastructure is often constrained by the availability of hardware resources and institutional capacity. These limitations reduce the number of students who can access the facilities simultaneously. Third, variations in software configuration across different devices can create inconsistencies in learning experiences, leading to additional technical challenges for students. [3], [4], [5]

These constraints directly affect the quality of programming education. Limited practice opportunities reduce student engagement and hinder skill development. In addition, delayed feedback in traditional laboratory settings slows the learning process. Students often need to wait for instructor evaluation, which interrupts the iterative cycle of coding, testing, and debugging. This condition is not aligned with the nature of programming learning, which requires immediate feedback and continuous experimentation.

The emergence of web-based technologies provides a promising solution to these challenges. Web-based learning systems enable users to access applications through standard browsers without the need for local installation. In the context of programming education, this capability allows the development of virtual programming laboratories that provide consistent and accessible coding environments. Students can write, compile, and execute code remotely, which significantly increases learning flexibility. [6]

Several studies have explored virtual laboratories and web-based programming environments. However, most existing works primarily focus on platform implementation, automated assessment, or remote access functionalities. Limited studies have integrated instructional materials, practical exercises, programming cheatsheets, and intelligent learning support within a single platform specifically designed for Information Systems education.

Furthermore, previous research generally emphasizes technical implementation while providing limited empirical evidence regarding learning effectiveness, usability evaluation, and pedagogical impact. Quantitative assessments using standardized instruments such as the System Usability Scale (SUS), ISO 25010 quality characteristics, and learning outcome measurements through pre-test and post-test experiments remain relatively scarce.

Therefore, this study addresses the identified gap by developing and evaluating a web-based virtual programming laboratory that combines practical programming environments with structured learning resources and learning support features.

The novelty of this research lies in three aspects: (1) Integration of programming practice, learning materials, cheatsheets, and intelligent suggestion features into a unified virtual laboratory platform. (2) Comprehensive evaluation using usability, software quality, and learning effectiveness measurements. (3) Empirical validation through pre-test and post-test experiments involving Information Systems students.

Virtual programming laboratories offer several advantages. They eliminate dependency on physical infrastructure, ensure uniform system environments, and support scalable access for multiple users. More importantly, they enable continuous learning by allowing students to practice programming at any time and from any location. The integration of

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

real-time feedback mechanisms further enhances learning effectiveness by enabling students to identify errors and refine their solutions immediately.

Despite these advantages, the development of a virtual programming laboratory requires careful consideration of both technical and pedagogical aspects. From a technical perspective, the system must ensure performance, scalability, and security, particularly when handling multiple concurrent users. From a pedagogical perspective, the system must support structured learning, provide meaningful feedback, and facilitate user interaction. Many existing studies focus on conceptual frameworks or prototype systems, which limits their applicability in real educational settings.

This research addresses these challenges by designing and developing a fully operational web-based virtual programming laboratory tailored for Information Systems education. The system has been implemented and deployed as a functional platform accessible online. It integrates key components, including an online code editor, execution engine, structured learning modules, and user management system. The integration of these components ensures a comprehensive learning environment that supports both students and instructors. [7], [8], [9]

The main contribution of this study is threefold. First, it provides a systematic design and implementation of a virtual programming laboratory based on software engineering principles. Second, it presents a fully deployed system that demonstrates practical feasibility and real-world applicability. Third, it evaluates the system in terms of usability and its impact on programming learning, providing empirical evidence of its effectiveness.

This paper is organized as follows. Section 2 describes the research methodology used in system development. Section 3 presents the results and discussion, including system implementation and evaluation. Section 4 concludes the study and outlines directions for future research.

2. RESEARCH METHODOLOGY

This study employs a structured methodology to design, develop, and evaluate a web-based virtual programming laboratory. The approach is grounded in software engineering principles and follows the Software Development Life Cycle (SDLC). The methodology ensures that the system is developed systematically, starting from problem identification to evaluation, while maintaining alignment with both technical requirements and educational objectives. [10]

The research adopts a design and development approach in which the main outcome is a functional system artifact. The study does not only analyze existing problems but also produces a solution in the form of a web-based virtual programming laboratory that can be directly used in educational settings.

The approach emphasizes iterative refinement. Each stage of development is evaluated and improved to ensure that the final system meets user expectations. The system serves as both a research output and a validation tool for assessing the effectiveness of virtual laboratories in programming education. [11], [12]

Requirement Analysis

The requirement analysis phase is conducted to identify system needs based on actual programming learning activities. The analysis considers user interaction, system functionality, and environmental constraints.

The system is designed to accommodate two primary user roles, namely students and instructors. Students interact with the system to perform programming activities, while instructors manage learning content and monitor student progress. Functional requirements define the core operations that the system must support.

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Table 1. Functional Requirements

No	Requirement	Description
1	User Authentication	Secure login with role-based access control
2	Code Editor	Interactive environment for writing and editing code
3	Code Execution	Server-side compilation and execution
4	Learning Modules	Structured programming materials
5	Output Feedback	Display execution results and error messages
6	User Management	Manage user roles and permissions

Table 2. Non-Functional Requirements

No	Requirement	Description
1	Usability	Simple and intuitive interface design
2	Performance	Fast response time during execution
3	Scalability	Ability to support multiple concurrent users
4	Security	Protection of user data and system access
5	Accessibility	Web-based access without installation

These requirements form the foundation for the subsequent design and implementation phases.

System Design

The system design phase transforms the identified requirements into a structured architecture and interaction model that aligns with the functional behavior illustrated in the system diagrams. The design emphasizes modularity, scalability, and separation of concerns to ensure that each component operates independently while maintaining seamless integration.

The system adopts a three-tier architecture consisting of presentation, application, and data layers. In addition, an execution layer is logically integrated within the application layer to handle code processing. This layered approach is consistent with the system architecture illustrated in Figure 4 and supports efficient communication between user interfaces and backend services.

The presentation layer is responsible for handling user interaction through a web-based interface. This layer is directly aligned with the interactions depicted in the use case diagram, where users perform activities such as authentication, accessing modules, and executing code. The interface provides structured navigation and interactive components that allow users to perform tasks efficiently. [13], [14]

The application layer processes user requests and controls system logic. This layer acts as the central coordinator that connects the presentation layer with the execution engine and data storage. Based on the activity diagrams, particularly the student and lecturer workflows, the application layer manages the sequence of operations starting from user input, validation, execution request handling, and response delivery. The separation of student and lecturer activity diagrams reflects role-based logic within this layer, where different permissions and processes are applied depending on the user role.

The execution engine operates as a specialized component responsible for compiling and executing programming code. Although logically part of the application layer, it functions as an isolated service to ensure security and performance. The activity diagram of the student illustrates this process clearly, where user-submitted code is sent to the execution engine, processed, and returned as output. This separation prevents direct exposure of system resources and ensures controlled execution.

The data layer is responsible for storing and managing system data, including user credentials, learning modules, and execution results. This layer supports persistent data

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

storage and ensures data consistency across system operations. It interacts with the application layer to retrieve and update information based on user actions.

The system behavior is modeled using Unified Modeling Language (UML) to ensure clarity in interaction design. The use case diagram, shown in Figure 1, illustrates the interaction between two primary actors, namely students and lecturers. Students interact with the system to perform activities such as logging in, accessing learning modules, writing code, and executing programs. Lecturers interact with the system to manage modules and monitor learning activities. These interactions define the scope of system functionality and ensure alignment with user requirements.

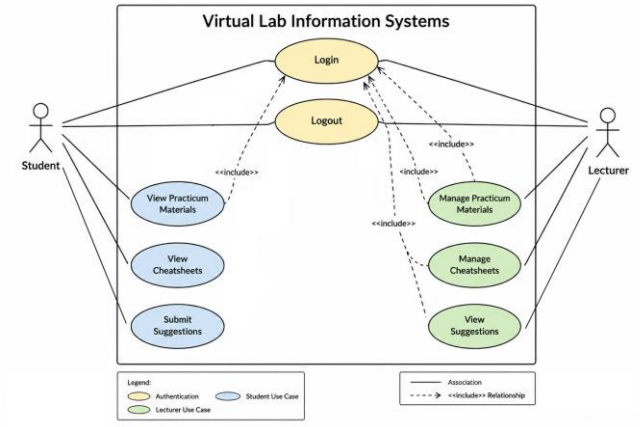

Figure 1. Use Case Diagram

The activity diagrams provide a more detailed representation of system workflows. The student activity diagram illustrates the process of code execution, starting from login, navigating to the code editor, writing code, submitting the code, and receiving execution results. This workflow reflects real-time interaction between the user interface and the execution engine.

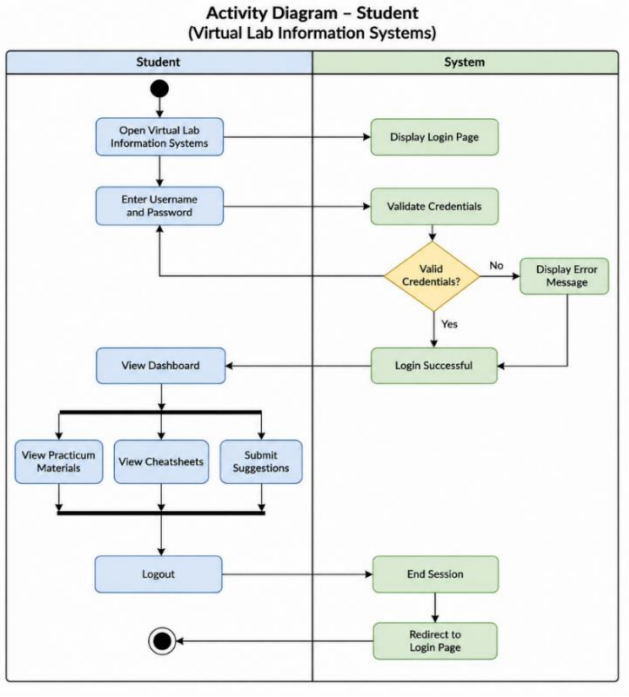

Figure 2. Activity Diagram Student

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The lecturer activity diagram represents a different workflow, focusing on content management and system monitoring. The process includes logging into the system, managing learning modules, and reviewing user activity. This distinction highlights the role-based design of the system.

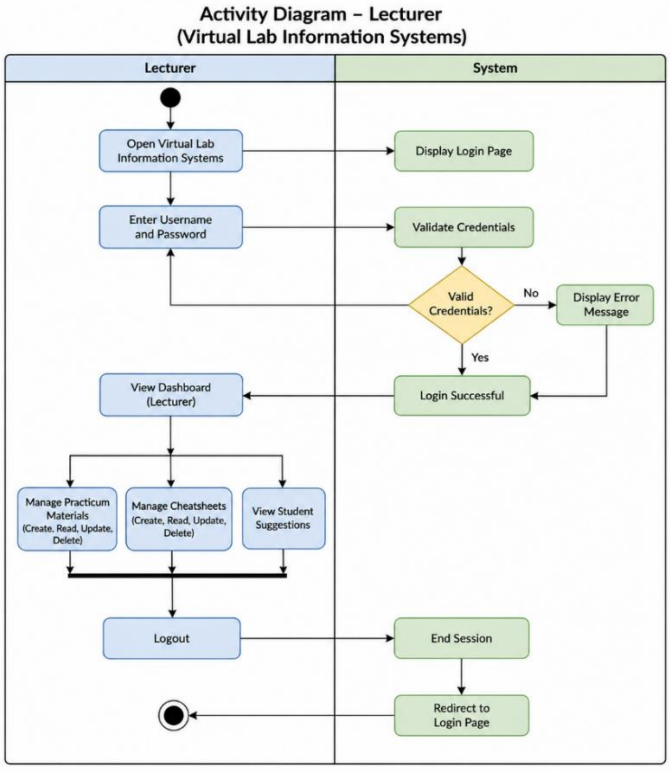

Figure 3. Activity Diagram Lecturer

The overall system architecture is illustrated in Figure 4, which shows the relationship between layers and system components. The diagram confirms that user requests originate from the presentation layer, are processed in the application layer, executed through the execution engine, and stored or retrieved from the data layer.

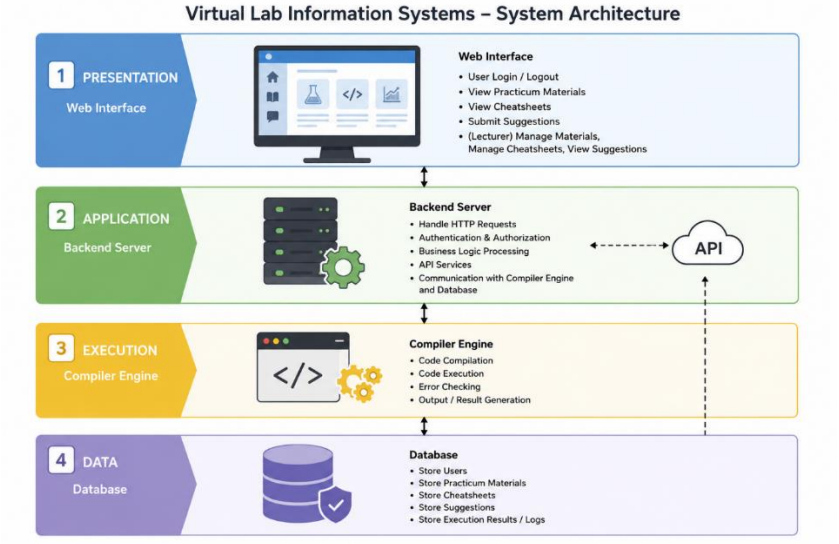

Figure 4. System Architecture

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The main components of the system architecture are summarized in Table 3.

Table 3. System Architecture Components

Layer	Component	Function
Presentation	Web Interface	Handles user interaction
Application	Backend Server	Processes logic and manages requests
Execution	Compiler Engine	Executes programming code
Data	Database	Stores user and module data

The alignment between system architecture and UML diagrams ensures that both structural and behavioral aspects of the system are clearly defined. The use case diagram defines system scope, the activity diagrams describe operational workflows, and the architecture diagram represents system structure. This integrated design approach ensures that the system is consistent, scalable, and capable of supporting real-time programming activities in an educational context.

System Implementation

The application layer is implemented as a backend server that processes user requests and manages system logic. This layer functions as the central controller that coordinates communication between the presentation layer, execution engine, and data layer. Based on the workflows defined in the activity diagrams, the application layer manages the sequence of operations, including request validation, execution routing, and response delivery. Role-based access control is enforced at this level to differentiate system behavior between student and lecturer roles, ensuring that each user interacts with appropriate functionalities.

The execution engine is implemented as a dedicated service responsible for compiling and executing programming code. Although logically integrated with the application layer, this component operates in an isolated environment to ensure execution security and resource management. When a user submits code through the web interface, the request is forwarded to the execution engine for processing. The engine compiles and executes the code, then returns the results, including standard output and error messages, to the application layer. The results are subsequently delivered to the presentation layer and displayed to the user. This execution flow directly corresponds to the process illustrated in the student activity diagram. [15], [16], [17]

The data layer is implemented using a database management system that stores user credentials, learning modules, execution history, and system logs. The backend server interacts with this layer to perform data retrieval and updates in response to user actions. This ensures data persistence, consistency, and integrity across all system operations.

The integration of these components enables real-time interaction between users and the system. The request–response mechanism ensures that code execution results are delivered immediately after processing. This capability is essential in programming education, as it allows students to receive instant feedback and iteratively refine their solutions.

From an architectural perspective, each layer performs a distinct function while maintaining efficient communication with other components. The presentation layer captures user input, the application layer processes system logic, the execution engine performs computational tasks, and the data layer manages persistent storage. This separation of concerns enhances maintainability, scalability, and system security.

Overall, the implementation successfully realizes the designed architecture and supports the workflows defined in the UML diagrams. The system provides a stable, interactive, and

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

scalable environment for programming learning, ensuring alignment between system design and operational behavior.

Testing Procedure

The testing phase is conducted to ensure that the system operates in accordance with the defined requirements and performs reliably under normal usage conditions. The testing strategy focuses on both functional correctness and system integration.

Functional testing is performed to verify that each feature operates as expected. This includes validation of user authentication, access to learning modules, code editing and execution, and output display. Each function is tested using predefined scenarios to ensure that the system produces correct and consistent results.

Integration testing is conducted to evaluate the interaction between system components. This process ensures that data flows correctly between the presentation layer, application layer, execution engine, and data layer. Particular attention is given to the communication between the backend server and the execution engine, as this interaction is critical for real-time code processing.

The testing results indicate that all major system functionalities operate correctly and consistently. No critical errors are identified during testing, and the system demonstrates stable performance during repeated usage. These results confirm that the system is reliable and suitable for deployment in real learning environments. [18], [19], [20]

Learning Experiment Design

To evaluate the educational effectiveness of the proposed virtual programming laboratory, a quasi-experimental study was conducted involving undergraduate students enrolled in an introductory programming course.

A total of 35 Information Systems students participated in the experiment. Before using the virtual laboratory, students completed a pre-test designed to measure their initial programming knowledge and problem-solving abilities. The students then utilized the virtual laboratory for four weeks during practical programming activities.

After the intervention period, a post-test containing questions of equivalent difficulty was administered. The improvement in learning outcomes was analyzed using a paired-sample t-test with a significance level of 0.05.

In addition to learning effectiveness evaluation, usability assessment was conducted using the System Usability Scale (SUS), while software quality was evaluated based on selected ISO 25010 characteristics.

Evaluation Method

The evaluation phase aims to assess the effectiveness of the system in supporting programming learning activities. The evaluation is conducted through direct interaction with the system in real usage scenarios, allowing assessment of both technical performance and user experience.

The evaluation focuses on several key aspects, including usability, accessibility, system performance, and learning support. Usability is assessed based on the ease of navigation, clarity of interface, and efficiency of user interaction. Accessibility is evaluated by the ability of users to access the system across different devices and environments without requiring additional configuration.

System performance is assessed in terms of responsiveness and execution speed, particularly during code processing activities. Learning support is evaluated based on how effectively the system facilitates programming practice, provides feedback, and supports iterative learning processes.

The evaluation results indicate that the system provides a positive user experience and effectively supports programming learning. Users are able to interact with the system

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

efficiently, and the real-time feedback mechanism enhances understanding and problem-solving ability. The findings also highlight areas for improvement, particularly in terms of scalability and feature expansion. [21], [22]

The methodology provides a comprehensive framework for developing and evaluating the virtual programming laboratory. It ensures that the system is not only technically robust but also aligned with the pedagogical requirements of programming education.

3. RESULT AND DISCUSSIONS

This section presents a comprehensive analysis of the implemented web-based virtual programming laboratory. The discussion covers system implementation, feature validation, performance behavior, user experience, and learning impact. The analysis is based on actual system deployment and real usage scenarios.

System Implementation and Deployment

The virtual programming laboratory has been fully implemented and deployed as a web-based system. The system operates through a centralized server architecture, allowing users to access all functionalities via a standard web browser. This approach eliminates the need for local software installation and ensures a consistent programming environment for all users.

The deployment process involves integrating the user interface, backend services, and execution engine into a unified platform. The system manages user requests dynamically, ensuring that each interaction is processed efficiently. The centralized nature of the system also simplifies maintenance and updates, as changes can be applied directly on the server without requiring user-side configuration.

From an operational perspective, the system demonstrates stability during continuous usage. Users are able to access the system, execute code, and retrieve results without interruption. This stability is essential for supporting learning activities that require uninterrupted interaction.

Comparison with Existing Virtual Lab Platforms

The comparison demonstrates that existing platforms mainly focus on code execution and programming challenges. In contrast, the proposed system integrates instructional content, practical exercises, lecturer management, and learning analytics within a single educational environment, making it more suitable for Information Systems education.

Table 4. System Architecture Components

Feature	Replit	JDoodle	HackerRank	Proposed System
Online Compiler	✓	✓	✓	✓
Practical Modules	✗	✗	Limited	✓
Cheatsheets	✗	✗	✗	✓
Lecturer Content Management	✗	✗	Limited	✓
Suggestion Feature	Limited	✗	✓	✓
OBE Assessment	✗	✗	✗	✓
Learning Analytics	✗	✗	Limited	✓
Information Systems Curriculum Alignment	✗	✗	✗	✓

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Interface Design and Feature Realization

The interface design of the virtual programming laboratory is developed to support effective interaction, reduce cognitive complexity, and align with programming learning workflows. Each interface component is designed based on specific user tasks and reflects the functional requirements defined in the system design. The system ensures consistency across interfaces, enabling users to transition smoothly between features.

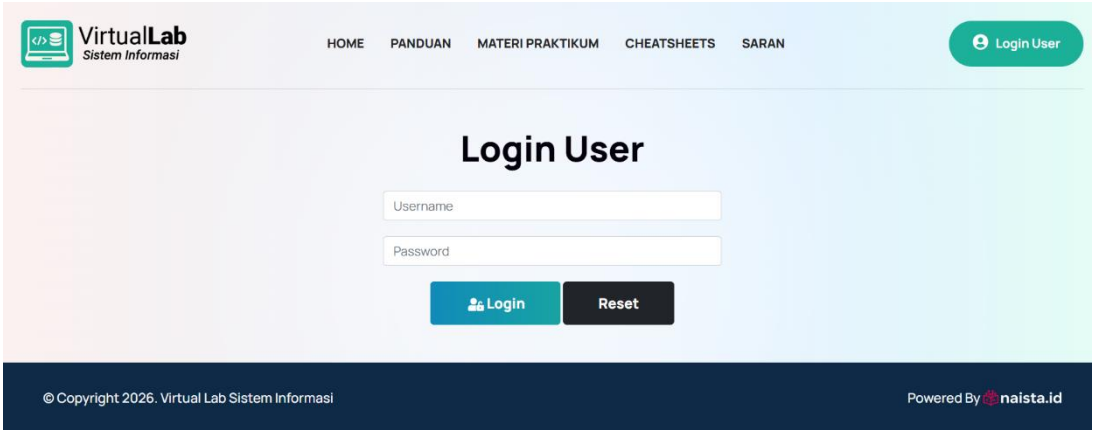

Figure 5. Login Interface

The login interface functions as the entry point of the system and implements secure authentication mechanisms. It includes input fields for user credentials and validation processes to ensure authorized access. The system applies role-based access control, where user roles such as student and lecturer determine subsequent system behavior. This interface ensures data security while maintaining ease of access.

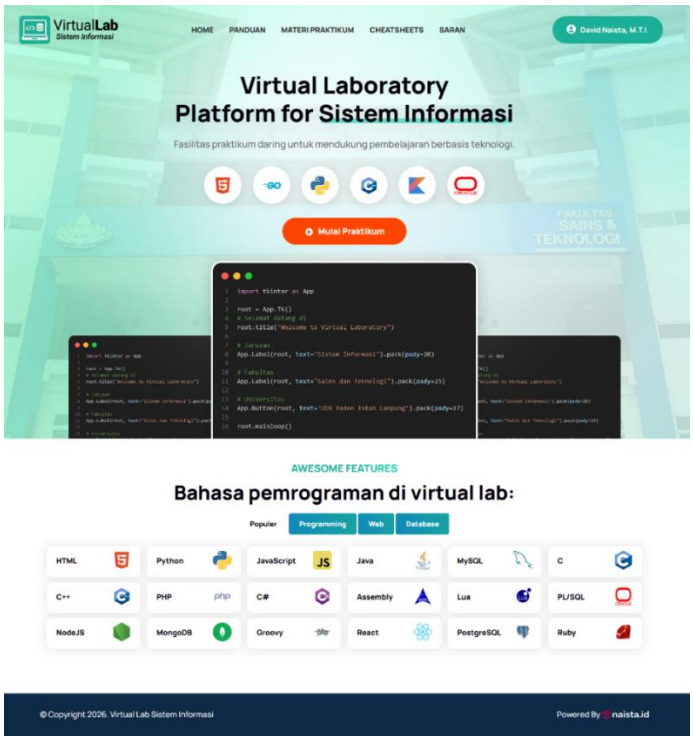

Figure 6. Homepage Interface

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The homepage interface serves as the central dashboard that integrates all system functionalities. It provides structured navigation elements, including menus for accessing the code compiler, learning materials, and supporting features. The layout prioritizes clarity and accessibility, allowing users to quickly identify available functions. This interface reduces navigation time and improves overall system usability.

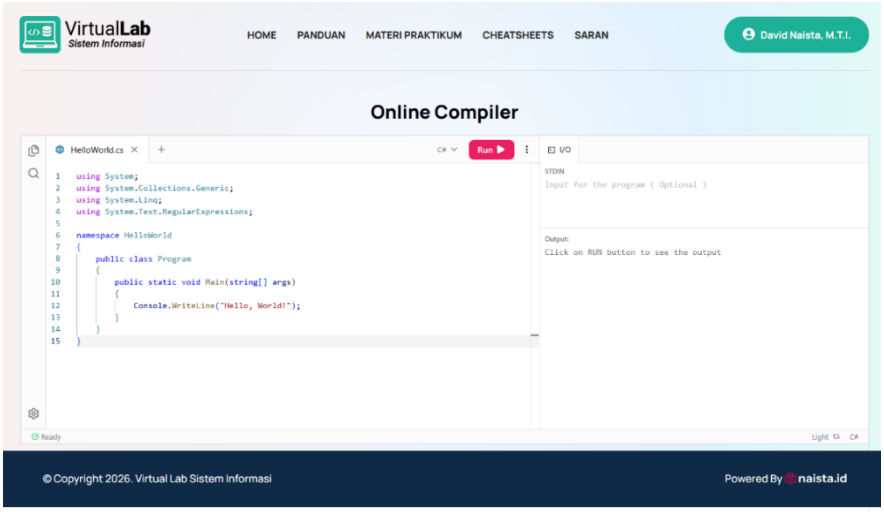

Figure 7. Code Compiler Interface

The code compiler interface represents the core operational component of the system. It integrates a code editor with an execution panel, enabling users to write, edit, and execute programming code in a single environment. The editor supports syntax highlighting and structured input, which enhances readability and reduces coding errors. The execution panel displays real-time output, including successful execution results and error messages. This direct interaction between input and output supports iterative learning and debugging.

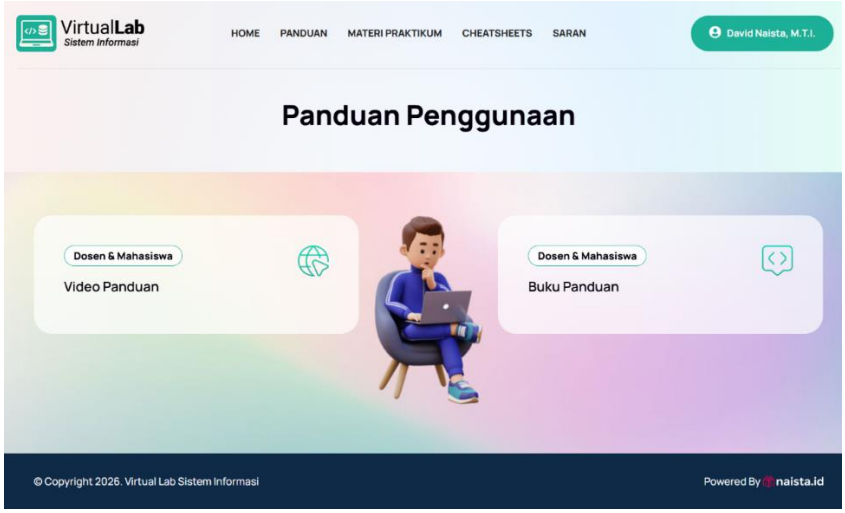

Figure 8. Guide Interface

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The guide interface provides instructional support that assists users in understanding both system usage and programming concepts. It contains structured explanations, usage instructions, and contextual guidance. This feature is particularly useful for beginner users, as it reduces dependency on external learning resources and supports independent learning.

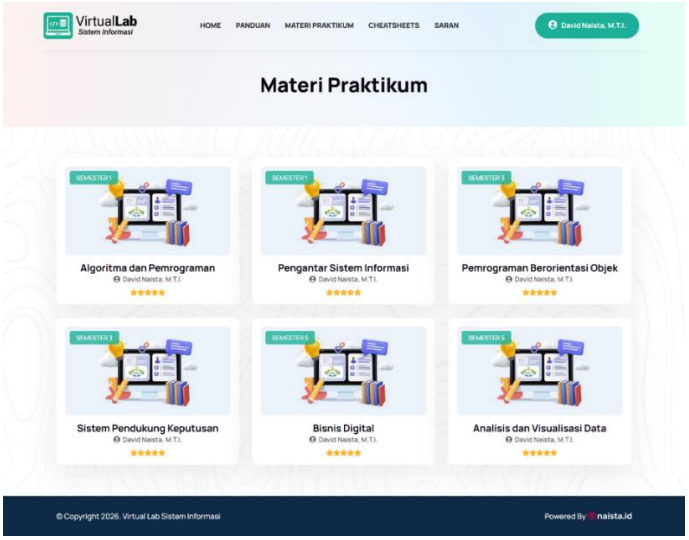

Figure 9. Practical Materials Interface

The practical materials interface contains structured programming exercises designed to support progressive learning. Each material is organized based on difficulty level or topic, allowing users to follow a systematic learning path. This interface plays a critical role in bridging theoretical knowledge and practical application.

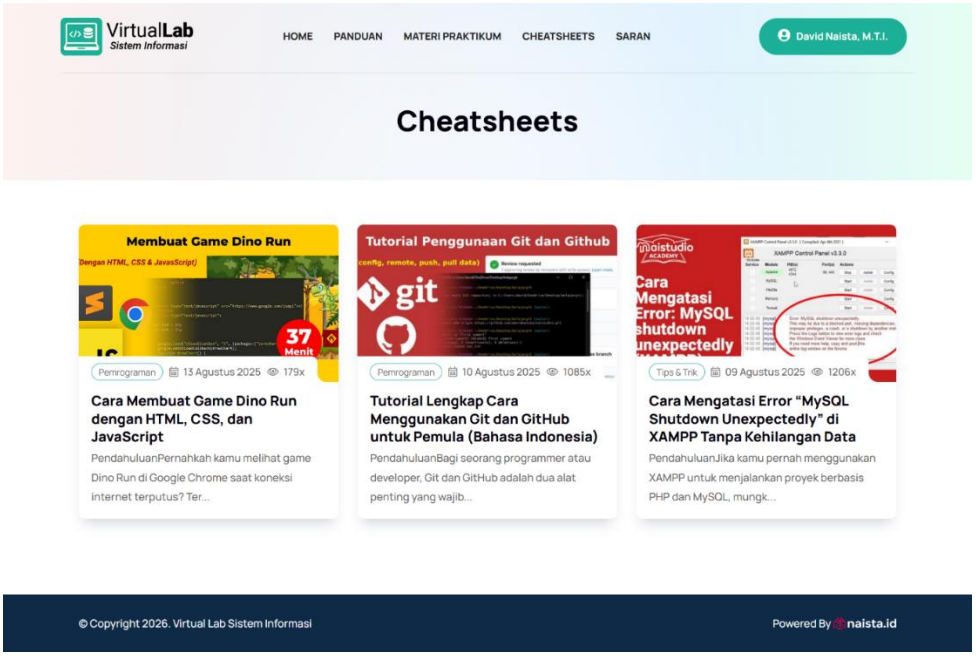

Figure 10. Cheatsheets Interface

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The cheatsheets interface provides concise and accessible references for programming syntax and concepts. It allows users to quickly retrieve essential information without interrupting their workflow. This feature improves efficiency during coding activities and reduces cognitive load associated with recalling syntax.

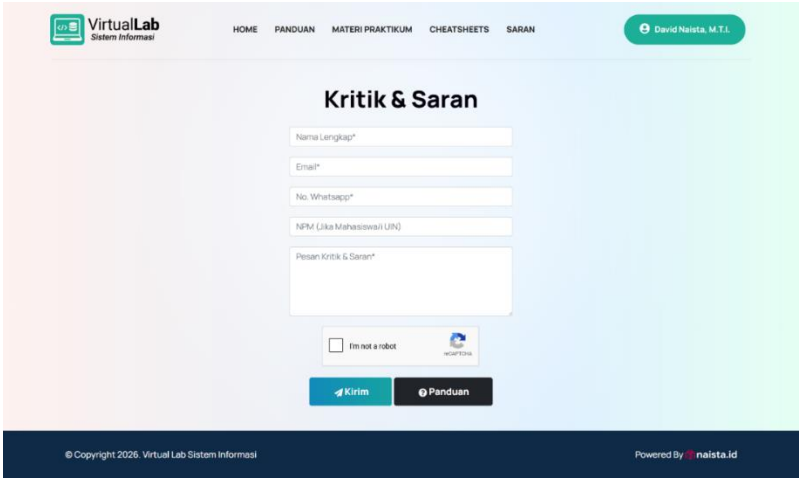

Figure 11. Suggestion Interface

The suggestion interface offers additional learning support by providing hints, recommendations, or guidance related to programming tasks. This feature enhances problem-solving by assisting users in identifying alternative approaches or correcting mistakes. It contributes to a more interactive and adaptive learning experience.

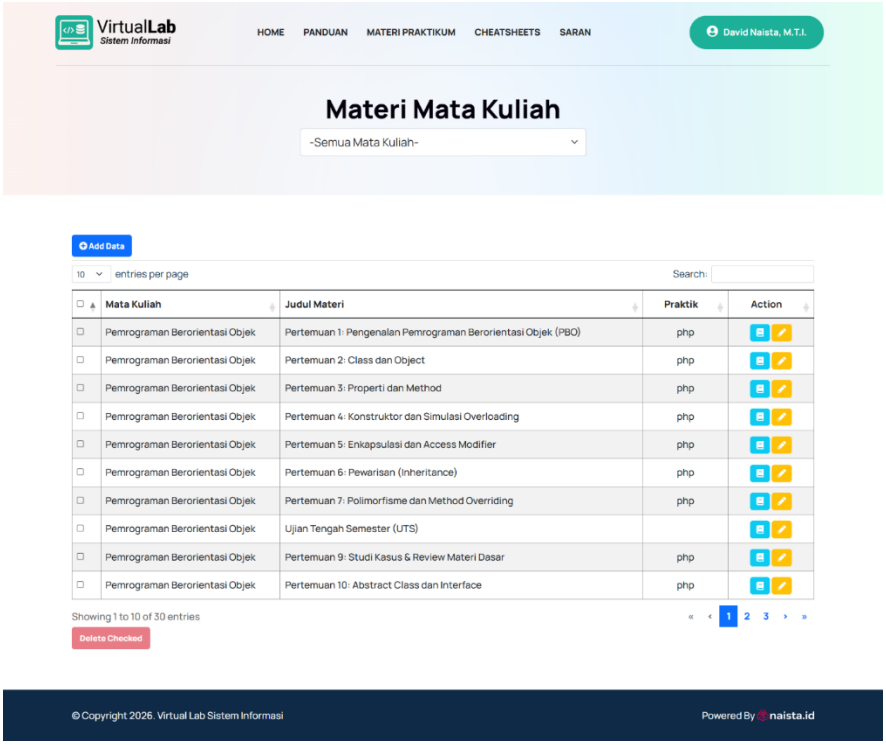

Figure 12. Practical Material Management

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

This interface is designed for lecturers to create and update learning materials. It includes input forms for adding content, defining tasks, and organizing instructional materials. The interface supports efficient content authoring and ensures that learning resources remain up to date.

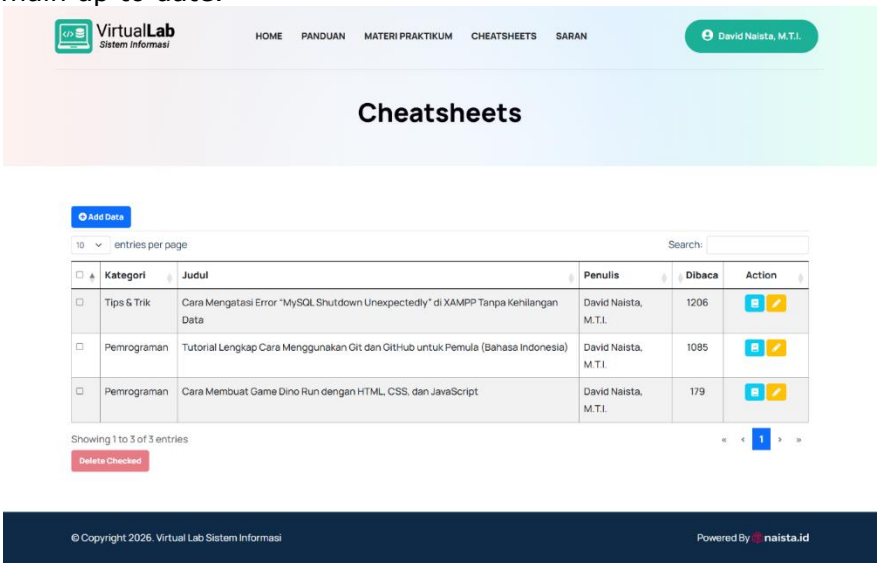

Figure 13. Practical Material Management

This interface allows lecturers to manage existing learning materials. It provides functionalities such as viewing, editing, and deleting content. The interface reflects administrative control and supports the lecturer's role in maintaining the learning environment.

The integration of these interfaces creates a unified system that supports both programming practice and instructional management. Each interface is designed with a clear functional purpose and aligned with user roles. The combination of coding tools, learning resources, and management features ensures that the system not only facilitates programming activities but also enhances the overall learning experience.

From a usability perspective, the system demonstrates consistency in layout, navigation, and interaction patterns. From a pedagogical perspective, the integration of guidance, practice, and feedback mechanisms supports active learning and continuous skill development. This alignment between interface design and learning objectives confirms that the system effectively addresses the needs of programming education. The alignment between system features and user needs is summarized in Table 5.

Table 5. Feature User Need Mapping

Feature	User Role	User Need	System Functionality
Login Interface	All Users	Secure system access	Authentication and role-based access control
Homepage Interface	All Users	Central navigation	Access to all system features
Code Compiler Interface	Student	Write and execute code	Integrated editor and execution engine
Guide Interface	Student	Learning assistance	Instructional guidance

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Practical Materials Interface	Student	Structured practice	Organized programming exercises
Cheatsheets Interface	Student	Quick reference	Syntax and concept summaries
Suggestion Interface	Student	Learning support	Recommendations and hints
Material Management Interface	Lecturer	Content management	Create and manage learning modules

This structured mapping demonstrates that the implemented system is not only technically functional but also aligned with the pedagogical requirements of programming education. Each interface component contributes to a cohesive learning experience by integrating practice, guidance, and content management within a single platform.

Functional Validation

The system is evaluated through functional testing to verify that all implemented features operate in accordance with the defined specifications. The testing process is designed to simulate real user interactions under normal operating conditions, ensuring that the system behavior reflects actual usage scenarios.

Functional validation focuses on assessing the correctness of core system functionalities, including user authentication, navigation, code compilation and execution, output visualization, and access to learning modules. Each function is tested using predefined scenarios that represent typical user activities, such as logging into the system, accessing instructional materials, writing and executing code, and reviewing execution results.

The testing process involves input validation, system response observation, and output verification. For each test case, expected outcomes are defined based on system requirements, and actual outcomes are compared to ensure consistency. This approach ensures that all system components operate reliably and produce accurate results.

In addition, the validation process examines the interaction between user roles, particularly students and lecturers. Role-based functionalities, such as accessing learning modules for students and managing instructional content for lecturers, are tested to confirm that access control mechanisms function correctly.

The results of functional testing indicate that all major features perform as expected without critical errors. The system successfully processes user inputs, executes programming code, and delivers accurate outputs in real time. The consistency of these results demonstrates that the system meets its functional requirements and is ready for deployment in real educational environments.

To summarize the testing outcomes, Table 6 presents the functional validation results for key system features.

Table 6. Functional Validation Results

No	Feature	Test Scenario	Expected Result	Actual Result
1	User Authentication	Login with valid credentials	Access granted	Successful
2	Homepage Navigation	Access main dashboard	Dashboard displayed correctly	Successful
3	Code Compilation & Execution	Run valid and invalid code	Output or error message displayed	Successful

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

4	Learning Module Access	Open and navigate modules	Content displayed correctly	Successful
5	Output Display	Show execution results	Correct output shown	Successful
6	Role-Based Access Control	Student vs lecturer access	Correct permission applied	Successful
7	Material Management	Create/edit/delete materials (lecturer)	Data updated correctly	Successful

The functional validation confirms that the system operates reliably and fulfills all required functionalities. These findings provide a strong foundation for further evaluation of system performance and user experience.

System Performance Analysis

System performance is evaluated based on response time, processing efficiency, and system stability during user interaction. The system adopts a request–response model, where user input is transmitted from the presentation layer to the application layer, processed by the execution engine, and returned as output to the user interface.

The response time remains within acceptable limits for interactive web-based systems, enabling smooth execution of programming tasks. The system consistently delivers outputs within a short time interval after code submission, which is critical in supporting real-time learning activities. Immediate feedback allows users to iteratively test and refine their code without disruption.

From an architectural perspective, performance efficiency is achieved through the separation of system layers. The distribution of responsibilities between the presentation layer, application logic, and execution engine minimizes processing overhead and reduces the risk of bottlenecks. The execution engine operates independently, allowing computational tasks to be handled without affecting user interface responsiveness.

The system demonstrates stable performance under normal usage conditions, including multiple simultaneous requests from users. However, performance limitations may emerge under high concurrency scenarios, particularly when multiple users execute complex code simultaneously. This indicates the need for further optimization strategies, such as load balancing, asynchronous processing, or container-based execution environments.

Usability and User Experience Evaluation

The system is evaluated from a usability perspective to assess the effectiveness of user interaction and overall experience. The evaluation is conducted through direct observation of user activities and qualitative feedback obtained during system usage.

The usability assessment focuses on key aspects, including interface clarity, navigation efficiency, interaction simplicity, and accessibility. The interface demonstrates a clear layout structure, allowing users to easily understand available features and system functions. Navigation between components, such as learning modules and the code compiler, is efficient and requires minimal effort.

User interaction is intuitive, enabling users to perform tasks such as writing and executing code without requiring extensive guidance. The system also supports cross-device accessibility, allowing users to access the platform through different devices and operating systems without compatibility issues. The results of the usability evaluation are summarized in Table 7.

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Table 7. Usability Evaluation

Aspect	Indicator	Evaluation Result
Interface Clarity	Ease of understanding layout	High
Navigation	Ease of moving between features	High
Interaction	Ease of performing tasks	High
Accessibility	Cross-device usability	Very High

The evaluation results indicate that the system provides a user-friendly and accessible environment. Users can interact with the system efficiently, which reduces learning barriers and enhances engagement.

Learning Effectiveness Analysis

The system contributes to improving programming learning outcomes through structured interaction, continuous accessibility, and real-time feedback mechanisms. The web-based nature of the platform enables students to access learning materials and practice programming at any time, extending learning activities beyond traditional classroom settings.

Real-time feedback is a critical factor in enhancing learning effectiveness. The system allows students to immediately identify errors in their code and understand program behavior. This immediate response supports iterative learning, where users continuously refine their solutions based on feedback.

The structured organization of learning modules further strengthens the learning process. Programming exercises are arranged in a progressive sequence, guiding students from basic concepts to more complex problem-solving tasks. This approach reduces cognitive overload and supports incremental knowledge acquisition.

The integration of supporting features, such as guides, cheatsheets, and suggestions, enhances the learning experience by providing contextual assistance. These features enable students to solve problems more effectively and develop independent learning skills.

Discussion

The findings demonstrate that the developed system effectively addresses key challenges in programming education, particularly in terms of accessibility, interactivity, and learning support. The web-based implementation eliminates infrastructure constraints and ensures a consistent programming environment for all users.

The integration of core components, including the code compiler, execution engine, and learning modules, ensures that both technical and pedagogical requirements are met. Unlike conceptual or prototype-based studies, this research presents a fully implemented system that has been deployed and tested in real usage scenarios. This practical implementation provides strong evidence of system feasibility and applicability in educational contexts.

Despite its strengths, the system has several limitations. The current implementation supports a limited number of programming languages, which may restrict its applicability across diverse programming courses. In addition, system performance under high user concurrency requires further optimization to maintain responsiveness and scalability.

Future development should focus on extending system capabilities by supporting additional programming languages, improving execution efficiency, and integrating intelligent feedback mechanisms. The incorporation of adaptive learning features, such as automated code analysis and personalized recommendations, can further enhance learning outcomes.

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

Overall, the system provides a scalable and effective solution for programming education. It aligns with current trends in digital and web-based learning environments and demonstrates strong potential for further development and wider adoption.

4. CONCLUSION

This study presents the design, development, and evaluation of a web-based virtual programming laboratory for Information Systems education. The research addresses key limitations of traditional programming laboratories, including restricted accessibility, dependency on physical infrastructure, and limited opportunities for continuous practice. The developed system leverages web-based technologies to provide a flexible and scalable learning environment that supports programming activities without spatial and temporal constraints.

The results confirm that the system successfully fulfills both functional and non-functional requirements. Core features, including the code editor, execution engine, output visualization, and structured learning modules, operate reliably and support real-time interaction. The system provides a stable and consistent programming environment, reducing technical barriers and ensuring uniform access for all users.

From a pedagogical perspective, the system enhances learning effectiveness by enabling continuous and self-directed practice. The real-time feedback mechanism allows students to identify errors immediately and refine their solutions through iterative processes. In addition, the structured organization of learning modules supports progressive learning, enabling students to build knowledge systematically from basic to advanced levels.

This study contributes by integrating software engineering principles with educational requirements and delivering a fully implemented system rather than a conceptual model. The availability of an operational platform strengthens the practical relevance of the research and demonstrates its applicability in real educational contexts. The system can be adopted by higher education institutions to improve accessibility, scalability, and the overall quality of programming education.

Despite these contributions, several limitations remain. The current system supports a limited range of programming languages, which may restrict its use across diverse curricula. System performance under high concurrency conditions also requires further optimization. In addition, the evaluation primarily relies on qualitative observations, which limits the generalizability of the findings.

Future research should focus on extending system capabilities, improving scalability, and incorporating quantitative evaluation methods. The integration of intelligent feedback mechanisms and adaptive learning features can further enhance personalization and learning outcomes. Expanding support for multiple programming languages and optimizing performance for large-scale deployment will also increase the system's applicability.

In conclusion, the developed virtual programming laboratory provides an effective, scalable, and practical solution for programming education in Information Systems. The findings demonstrate that web-based virtual laboratories can significantly enhance accessibility, support continuous learning, and improve student engagement, making them a viable alternative to traditional laboratory environments.

5. ACKNOWLEDGMENT

The author would like to express sincere gratitude to the Information Systems Study Program, Faculty of Science and Technology, UIN Raden Intan Lampung, for providing institutional support throughout the development of this research. Appreciation is also extended to colleagues and students who participated in the testing and evaluation process of the virtual programming laboratory.

David Naista: *Coresponding Author

CHAIN: Journal of Computer Technology, Computer Engineering and Informatics
Website : https://ejournal.techcart-press.com/index.php/chain
E-ISSN 2964-2485
P-ISSN 2964-2450
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/chain.v4i3.294**
Page 251-269

The development and implementation of the system would not have been possible without the valuable feedback and constructive input from users during the trial phase. Their contributions have played an important role in improving the functionality and usability of the system.

The author also acknowledges all researchers and practitioners whose works have contributed to the theoretical and technical foundations of this study.

6. REFERENCES

[1] R. Raman, K. Achuthan, V. K. Nair, and P. Nedungadi, "Virtual Laboratories: A historical review and bibliometric analysis of the past three decades," *Educ. Inf. Technol.*, 2022, doi: 10.1007/s10639-022-11058-9.
[2] M. Messer, N. C. C. Brown, M. Kolling, and M. Shi, "Automated Grading and Feedback Tools for Programming Education: A Systematic Review," *ACM Trans. Comput. Educ.*, 2024.
[3] S. Yang, M. Baird, and E. O'Rourke, "Debugging Interventions in Programming Education," *ACM Trans. Comput. Educ.*, 2024.
[4] A. Espinal, C. Vieira, and A. J. Magana, "Professional Development in Computational Thinking," *ACM Trans. Comput. Educ.*, 2024.
[5] A. Alammary, "Blended Learning Models for Introductory Programming Courses," *PLoS One*, 2019.
[6] I. Sommerville, *Software Engineering*. Pearson, 2021.
[7] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2020.
[8] J. M. Wing, "Computational Thinking's Influence on Research and Education," *Ital. J. Educ. Technol.*, 2020.
[9] H. Keuning, J. Jeuring, and B. Heeren, "Code Quality in Programming Education," *Comput. Sci. Educ.*, 2024.
[10] K. Ishaq and A. Alvi, "Personalized Learning in Programming Education," *PeerJ Comput. Sci.*, 2024.
[11] S. Sentance and A. Waite, "PRIMM: Exploring Pedagogical Approaches for Teaching Programming," *Comput. Sci. Educ.*, 2021.
[12] A. Robins, "Novice Programmers and Introductory Programming," *Comput. Sci. Educ.*, 2019.
[13] T. Clear, "Learning to Program: A Review," *Comput. Sci. Educ.*, 2020.
[14] L. Thomas and M. Ratcliffe, "Feedback in Programming Education," *IEEE Trans. Educ.*, 2022.
[15] K. Falkner and R. Vivian, "Systems Supporting Programming Learning: A Survey," *ACM TOCE*, 2022.
[16] M. Tawafak and A. Romli, "E-Learning Systems Review," *Educ. Inf. Technol.*, 2021.
[17] M. Al-Samarraie, "Cloud Learning Tools," *Educ. Inf. Technol.*, 2021.
[18] H. L. Llorens-Largo, "Web-Based Platform for Programming Learning," *IEEE Trans. Learn. Technol.*, 2021.
[19] S. Papadakis, "Coding Applications and Computational Thinking," *Educ. Inf. Technol.*, 2021.
[20] N. Nguyen, "Virtual Laboratories in Higher Education," *IEEE Access*, 2022.
[21] R. Anderson, "Scalable Online Programming Environments," *IEEE Softw.*, 2022.
[22] L. Malmi, "Computing Pedagogies Review," *arXiv*, 2024.

David Naista: *Coresponding Author