Edustruct: Development of a Web-Based Platform for Visualizing Basic Data Structures to Support Student Learning

Arieni Zahira Nur Firmansyah[a], Bagas Yogta Nugraha[b], Rajib Jabar Nur Furqon[c], Nusa Jati Gofur[d], Annuar Hakim[e], Muhammad Taufik Dwi Putra[f]
[a,b,c,d,e,f] *Computer Engineering Study Program, Indonesia University of Education*
Corresponding Author:
[f]*tdputra@upi.edu*

ABSTRACT
This study aims to develop EduStruct as a web-based learning tool designed to help students understand data structure concepts visually and interactively. The background of this study is the difficulty students face in understanding abstract data structures when learning them solely through theory without visualization. This study employs a Research and Development (R&D) approach using a prototype development model. The development process consisted of the following stages: requirements analysis, system design, implementation, and functional testing. The EduStruct system includes interactive visualizations of data structures such as arrays, stacks, queues, and sorting algorithms. The results of the development show that the EduStruct system was successfully implemented and all features function as designed. System testing was conducted using the black-box testing method to ensure that every data structure operation runs properly and responds effectively to user interactions. Thus, the EduStruct system is functioning well and is ready to be implemented as a visual-based learning tool in data structures courses.

Keywords : Data Structures, Visualization, Interactive Web.

INTRODUCTION

Data structures are one of the fundamental topics in computer science that form the basis for algorithm development and programming. Concepts such as arrays, stacks, queues, and sorting algorithms are essential foundational topics that computer science students must understand, as they play a role in developing logical and computational thinking skills. In the world of computing, effective data management has become crucial as the volume of data continues to grow and become increasingly complex. Therefore, the concept of data structures is highly relevant because it pertains to how data is organized, stored, and accessed efficiently within computer systems (Ginting et al., 2023).

However, the process of learning data structures is often perceived as difficult by beginner students. This is due to the abstract and dynamic nature of data structure concepts, making them hard to grasp through theoretical explanations or static representations such as text and images alone. Operations such as insertion, deletion, and reordering of data are often difficult for students to visualize. Difficulty in understanding basic data structures is also one of the challenges in learning programming.

With technological advancements, visualization-based learning approaches have become increasingly popular to aid in understanding algorithmic and data structure concepts.

Visualization allows abstract processes to be presented in an interactive format, enabling users to observe data changes in real time. This has been shown to enhance understanding by reducing cognitive load during the learning process.

Nevertheless, most existing visualization platforms still have several limitations, such as the use of English, complex interfaces, and a lack of focus on basic data structures for beginners. This aligns with findings from previous research highlighting that the main difficulty students face in learning data structures is understanding basic logical flow (Sholeh & Rachmawati, 2022; Dharma, 2018). This situation can hinder early-level students from effectively grasping the fundamental concepts of data structures.

Based on these issues, this study developed EduStruct, a web-based visual data structure simulation platform designed to help students understand the basic concepts of data structures through an interactive and simple approach. The system focuses on basic data structures such as arrays, stacks, queues, and sorting algorithms by displaying the process step by step (step-by-step visualization).

EduStruct was built using modern web technologies based on React and Vite with a Single Page Application (SPA) approach. The system also uses interactive animations to illustrate data changes in real time, allowing users to understand algorithmic processes more intuitively.

With EduStruct, it is hoped that the process of learning data structures will become easier to understand, more interactive, and better suited to the needs of beginner students in learning the fundamentals of algorithms and data structures.

LITERATURE REVIEW

Data Structures

Data structures are a way to group and organize data into a specific form so that data can be stored, accessed, and managed more efficiently. Data structures consist of a collection of variables defined as a single unit, where each variable can have a different data type depending on data processing needs (Soetanto, 2022; Winarsih & Wahono, 2022).

Array

An array is a data structure consisting of a collection of elements of the same data type stored within a single variable. Each element can be accessed using an index, making the data processing process more structured and efficient (Putri et al., 2024). An array has a fixed number of elements and can be one-dimensional, two-dimensional, or multidimensional. Additionally, an array can only store data of a uniform type, such as integers, floats, characters, or strings.

Sorting Algorithms

Sorting is the process of arranging data in a specific order such as from smallest to largest or vice versa to make the data easier to manage and search. In the Algorithms and Data Structures course, sorting is one of the essential foundational topics because it plays a key role in improving the efficiency of data processing and retrieval. The sorting process is widely used in various scenarios, such as sorting student data, transactions, or index-based

information (Bawamenewi et al., 2022; Iskandar et al., 2023). Some basic sorting algorithms commonly used in education are Bubble Sort, Selection Sort, and Insertion Sort because they have simple, easy-to-understand workflows. Each algorithm has different mechanisms and performance characteristics depending on the conditions of the data being processed (Gustina et al., 2025).

Queue

A queue is a linear data structure that follows the First In First Out (FIFO) principle, meaning that the element added first will be the first element removed. The main operations on a queue consist of enqueue, which adds data to the back of the queue, and dequeue, which removes data from the front of the queue. This data structure is widely used in various service systems because it maintains a regular and fair order of processes (Dinanti et al., 2025; Nasir et al., 2024).

Stack

A stack is a linear data structure that follows the Last In First Out (LIFO) principle, meaning the last element added is the first to be processed or removed. The main operations on a stack are `push`, which adds data to the top of the stack, and `pop`, which removes data from the top of the stack. This data structure is frequently used in programming processes that require a stacked data storage mechanism (Simbolon et al., 2024).

Interactive Media

Interactive media is a learning medium that allows users to actively participate in the learning process through various features such as simulations, exercises, interactive choices, and immediate feedback. This medium not only presents information passively but also provides users with the opportunity to interact with the learning material, making the learning process more engaging and effective. The use of interactive media can increase user engagement in understanding the material because it supports more dynamic and communicative learning activities (Jafnihirda et al., 2023).

State-Driven Rendering

Rendering on a website is the process of displaying a web page in a visual form that users can view and interact with through a browser. This process is carried out by combining various components such as HTML, CSS, JavaScript, and data from the back end to produce an interactive user interface (Ardiyanto & Ardhianto, 2024). In modern web application development, the rendering process can be performed using a state-driven rendering approach, a mechanism in which the user interface is automatically updated based on changes in the system’s state or data. Every data change triggers a re-render process for the relevant components, allowing the visualization to be displayed in real time without the need for a full page reload. This approach is widely used in frameworks such as React.js because it produces applications that are more responsive, dynamic, and interactive for users.

Animation in Learning

The use of animations in educational media can help improve users' understanding of the material being presented because information is displayed in a more engaging and interactive way. Animations make the learning process more dynamic, allowing users to more easily grasp the flow of the material and increasing engagement throughout the learning process. Additionally, the use of interactive visual elements in web-based learning applications can create a more effective learning experience compared to the static presentation of material (Tanjung et al., 2025).

Algorithm Visualization

Advances in information technology offer significant opportunities for developing more interactive and effective learning media. One widely adopted approach is algorithm visualization, a method that visually displays an algorithm's operation so users can more easily understand the workflow and changes in data. This approach helps make abstract algorithmic concepts more concrete and easier to understand. Furthermore, the use of modern web technologies enables the development of responsive and interactive visualization applications (Triyono et al., 2024).

Web-Based Learning Visualization

Learning abstract concepts often poses a challenge for students, especially when the material requires the ability to visualize processes dynamically. The lack of interactive media in learning makes it difficult for students to accurately understand conceptual representations and process flows. Therefore, the use of web-based visualization can help provide more concrete and interactive representations, thereby facilitating the process of conceptual understanding (Suryani et al., 2026). Research by Suryani et al. (2026) developed web-based visualizations to aid in calculus learning through interactive representations of rotating objects. The results showed that web-based visualizations can help users understand abstract concepts more concretely. However, this study focused on calculus and did not address data structure visualizations.

React.js

React.js is an open-source JavaScript library used to build dynamic and interactive user interfaces. React.js employs a component-based approach, allowing the application's UI to be divided into small, reusable parts that can be managed separately. This approach helps developers streamline the process of developing and maintaining web applications. React.js also supports the development of Single-Page Applications (SPAs) with fast and responsive performance through the use of a virtual DOM. This technology allows the view to be updated efficiently, updating only the parts that have changed. Additionally, React.js is declarative and flexible, making it widely used in modern web application development to create more interactive and efficient interfaces (Nasution & Iswari, 2021).

Interactive Web Application Architecture

A Single-Page Application (SPA) is a web application concept that loads only one main page and dynamically updates its content without a full page reload. This approach enables faster and more responsive navigation because UI changes occur immediately in response to user interactions (Santoso, 2021; Bismoputro et al., 2024). Furthermore, SPAs improve data delivery efficiency and provide a more interactive user experience compared to the traditional Multi-Page Application (MPA) approach.

Black Box Testing

Black Box Testing is a system testing method that involves testing an application's functions and features without examining the underlying source code. This testing aims to determine whether the system operates as intended, including input processes, application menus, and system outputs. In this method, testing is performed using various data inputs to ensure the system can accept valid data and reject invalid data. Thus, Black Box Testing is used to identify system functional errors and ensure that all application features operate in accordance with user needs and the designed specifications (Umiringsih et al., 2022; Maspupah, 2024).

Existing Algorithm Visualization Platforms

Numerous algorithm visualization platforms have been developed to facilitate the interactive learning of algorithms and data structures. One well-known platform is VisuAlgo, which provides simulations of various algorithms such as sorting, arrays, trees, and graphs through step-by-step animations. The VisuAlgo platform features a very minimalist interface; however, to use its features, users must understand some basic programming logic. In addition, there are several other tools, such as AAnim, Galant, HTML5 and JavaScript-based visualizations, that are used to display algorithmic processes dynamically and interactively. Algorithm visualization has been shown to help students understand changes in data structures and algorithmic flow more easily than text-based learning alone (Maravas & Pantouvakis, 2025).

Nevertheless, most existing visualization platforms still have several limitations, such as being in English, having interfaces that are relatively complex for beginner users, and lacking a focus on basic data structures like arrays, stacks, and queues. Furthermore, some platforms are geared more toward exploring advanced algorithms, making them less suitable as a basic learning tool for beginner students. Therefore, a simpler, more interactive, and easier-to-understand learning platform is needed to support the learning process for basic data structures.

Research Gap

Based on previous research, most algorithm visualization platforms still focus on advanced algorithms and use interfaces that are relatively complex for beginner users. Furthermore, the use of English and the lack of focus on basic data structures mean that these learning tools are not yet fully suitable for use by first-year students. Therefore, a simpler, more interactive, and

easier-to-understand web-based visualization platform is needed to support the learning of basic data structures.

Research Position (EduStruct)

Based on these gaps, this study developed EduStruct, a web-based visual data structure simulation platform that focuses on basic data structures such as arrays, stacks, queues, and sorting algorithms. EduStruct is designed with a simple, interactive approach and uses the Indonesian language to facilitate understanding among beginner students. This system displays data structure processes step-by-step (step-by-step visualization) so that users can understand data changes more intuitively. Thus, EduStruct is expected to serve as an alternative learning medium that is more accessible, simpler, and better suited to the needs of beginning students in learning data structures.

METHODS

Research Approach

This study employs a Research and Development (R&D) approach with the goal of developing a web-based learning platform that can help students understand data structure concepts visually and interactively.
The R&D approach was chosen because the research focuses not only on analysis but also on product development in the form of the EduStruct system, which can be used directly as a learning tool.

System Development Method

The development of the EduStruct system uses the Prototype Development Model, which consists of the following stages:



Figure 1. Diagram of the Prototype Development Model

1. Requirements Analysis
   In this stage, system requirements are identified based on problems encountered in learning data structures. The analysis results show that students have difficulty understanding abstract concepts such as arrays, stacks, queues, and sorting when using only conventional learning methods.
2. System Design
   During the system design phase, the architecture of the EduStruct application was developed, which included designing the page structure for the array, stack, queue, and sorting modules; designing the user interaction flow; and designing step-by-step visualizations of data changes. The system was designed using the Single Page Application (SPA) concept so that navigation between pages can be done quickly without the need to fully reload the page. This approach allows the application to

provide a more responsive and interactive user experience in the process of learning data structures (Santoso, 2021).

3. Data Collection Techniques

   The data in this study were obtained through two data collection techniques: a literature review and structured interviews. The literature review involved examining books, journals, similar applications, and scientific proceedings related to data structures, algorithm visualization, and web-based interactive learning media. The results of this review served as both a theoretical foundation and a basis for formulating system requirements.

   Interviews were conducted with 10 Computer Engineering students who had taken the data structures course online at the UPI Cibiru campus of the Indonesia University of Education. Each student was given 9 short questions in the same order. Questions asked before using the application inquired about difficulties encountered in understanding data structure concepts, such as the operation flow of stacks and queues. Questions asked after using the application inquired whether those difficulties had decreased and which parts of the application were helpful or still confusing. Answers were recorded by filling out a short questionnaire provided for this purpose.

   The interview responses were analyzed using qualitative descriptive methods by grouping answers based on difficulty topics and comparing responses before and after using the application. Responses before using the application were used to identify students' difficulties, which formed the basis for feature requirements, while comparisons with responses after using the application were used to assess the system's benefits.

4. System Implementation

   During the implementation phase, the EduStruct system was developed using several modern web technologies to support the application's performance and interactivity. React.js was used as the primary framework for building a component-based interface so that each feature could be managed modularly and efficiently (Nasution & Iswari, 2021). Vite was used as a build tool to accelerate the application's development and rendering processes. Additionally, React Router DOM was utilized to manage navigation between data structure module pages without the need to reload the page. To enhance visualization, the system uses Framer Motion to display dynamic and interactive data change animations, while Tailwind CSS was used to build a responsive, simple, and user-friendly interface. Each data structure is implemented as a separate component according to its function, such as the Array component for manipulating data indices, the Stack component for push and pop operations, the Queue component for enqueue and dequeue processes, and the Sorting component for simulating the step-by-step data sorting process.

5. Data Structure Logic Implementation

   Each data structure is represented using React's state (useState) to store data dynamically.

Table 1. Data Structure Logic

	Function
Array	Manipulating elements based on index
Stack	Uses the Last In First Out (LIFO) principle
Queue	Uses the First In First Out (FIFO) principle
Sorting	Using basic algorithms such as bubble sort in a step-by-step manner

6. Visualization Implementation
   Changes in the system's state will automatically trigger the re-rendering of components so that data visualizations can be displayed in real time. EduStruct employs a state-driven rendering approach, meaning that any data changes are immediately updated on the user interface without the need to reload the page. This approach allows for the dynamic and interactive visualization of algorithms and data structures.
   To clarify the data change process, the system uses transition animations for each operation, such as the movement of elements in a stack, the movement of elements in a queue, and the swapping of element positions during the sorting process. This visualization aims to help users understand the algorithm's workflow and changes in data structures step by step, making abstract concepts easier to grasp.
   System analysis was conducted descriptively by reviewing the alignment of system functions with data structure concepts, the effectiveness of visualizations for each data operation, and the system's response to user interactions. This analysis aims to ensure that the system can accurately and interactively represent data structure processes.
   System testing was conducted using the Black Box Testing method to ensure that all application functions operate in accordance with the designed requirements and specifications (Uminingsih et al., 2022). Testing was performed on various key features, such as data input and manipulation, navigation between pages, data structure simulation, data change animations, and the system's response to user interactions. During the testing, the tester entered specific input data into the system and then observed the actual responses or results displayed by the EduStruct interface. These actual results were then compared with the expected results. The test was deemed successful if all system functionalities responded to the input in accordance with the design specifications without generating any error messages during the animations.

RESULTS

Implementation of the EduStruct System:

This research produced a web-based learning platform called EduStruct, designed to interactively visualize basic data structures. The system was successfully implemented using React.js and Vite with a Single Page Application (SPA) architecture, allowing all user interactions to occur without reloading the page. EduStruct provides four main modules:

Array, Stack, Queue, and Sorting. Each designed to represent data structure operations visually and step-by-step. This approach allows users to observe changes in data state directly, thereby strengthening the connection between theoretical concepts and visual representations.



Figure 2. EduStruct Main Page

Implementation Results of the EduStruct Array Module

The Array module in EduStruct serves as a representation of a linear data structure that allows for dynamic manipulation of elements based on their indices. Implemented features include adding elements, updating values, deleting elements by index, and resetting data. The implementation results show that every operation on the array can be visualized in real time through a state-driven rendering approach. Changes in the data state automatically trigger a re-render of the interface components, so that every change to the array structure is immediately reflected on the screen without reloading the page. This approach offers the following benefits:

1 Improved system responsiveness,
2 Instant visualization of data changes,
3 And a better understanding of the concepts of indexing and element manipulation.

Table 2. Functional Testing Results for the Array Module in EduStruct

No	Feature	Test Description	Results
1	Create Array	Create an array with a size of 1–20	Success
2	Add a Single Value	Adding a single element to the array	Success
3	Add Batch Value	Add multiple elements at once	Success
4	Update Value	Change a value based on its index	Success
5	Delete Value	Delete an element based on its index	Success
6	Clear Array	Delete all array elements	Success

7	Random Generator	Automatically generates an array	Success

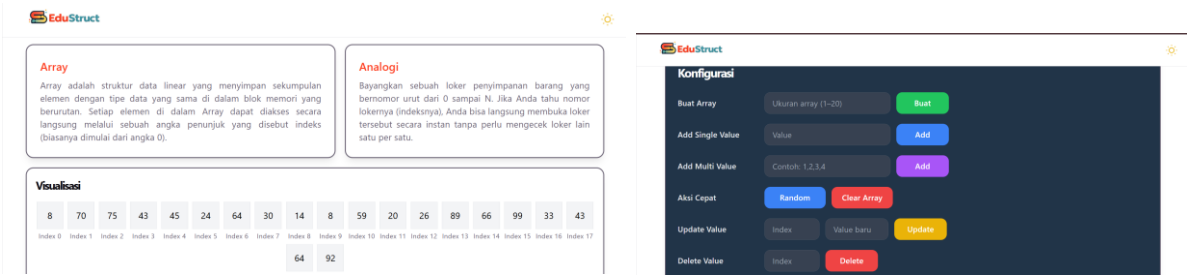


Figure 3. Visualization of the Array module in EduStruct (a) Array Module Display (b) Insert and Update Operations in the Array Module

Results of the EduStruct Sorting Module Implementation

The Sorting Module results represent the data sorting process using the Bubble Sort, Selection Sort, and Insertion Sort algorithms with an animation-based visual approach. The sorting process is visualized as bar elements whose heights change according to the data values. Each iteration of the algorithm is represented as a separate state, allowing users to observe:

1 The elements currently being compared (*active comparison*)
2 The value exchange process (*swap operation*)
3 Changes in the data order at each iteration
4 The final state when the data is sorted

The implementation results show that the *step-tracing visualization* approach makes the complexity of the sorting algorithm easier to understand intuitively.

Table 3. Functional Testing Results for the Sorting Module

No	Features	Test Description	Results
1	Data Input	Batch data import	Successful
2	Random Array	Automatically generate random data	Successful
3	Bubble Sort	Bubble Sort Algorithm	Successful
4	Selection Sort	Sorting using the Selection Sort method	Successful
5	Insertion Sort	Sorting using the Insertion Sort method	Successful
6	Ascending Sort	Sorting from smallest to largest	Successful
7	Descending Sort	Sorting from largest to smallest	Successful
8	Step Visualization	Animation of the sorting process step by step	Successful

9	Speed Control	Animation speed settings	Successful
10	Reset Array	Clear all data	Successful

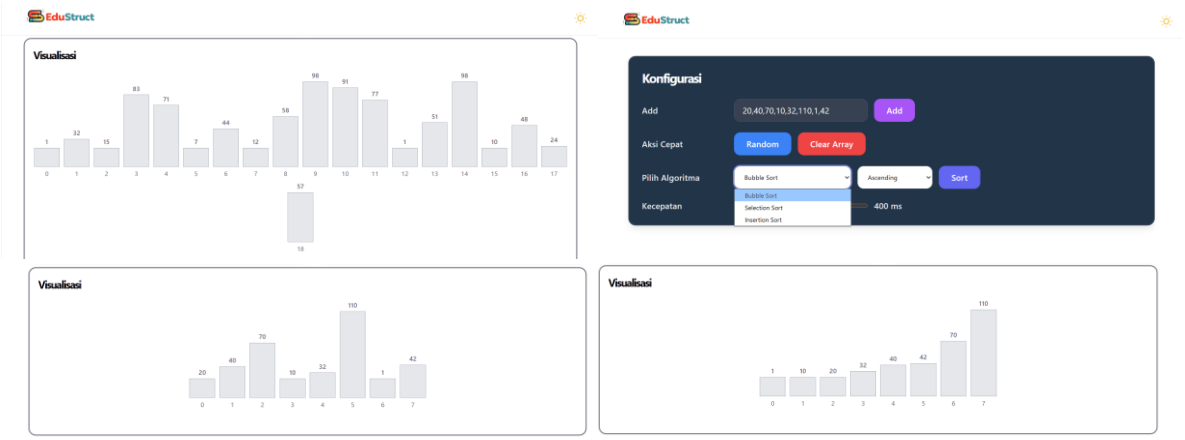


Figure 4. Visualization of the Sorting Module (a) Initial Display of the Sorting Module in EduStruct (b) Input of Array Values/Data (c) Results After Data/Array Input (d) Sorting Results

Results of the EduStruct Queue Module Implementation

The Queue Module represents a linear data structure based on the First In First Out (FIFO) principle. The implementation includes enqueue and dequeue operations, as well as element priority management.

Each element is visualized as a sequential node indicating its front and rear positions. The system automatically updates the state with every operation, allowing changes in the queue to be observed in real time.

Each element is visualized as a sequential node indicating the front and rear positions. The system automatically updates the state with every operation, allowing changes to the queue to be observed in real time. Additionally, this module includes a Priority Queue feature.

Through the Priority Index, users can assign a priority value to an element upon insertion. When Priority Dequeue is executed, the element with the highest priority is removed first. The system clarifies that the fundamental FIFO principle is not lost but remains in effect as a secondary rule for sorting elements with the same priority value.

The implementation results show that:

1 Elements are always added to the rear,
2 Elements are removed from the front,
3 And priority elements can be accessed via a specific index.

This visualization reinforces the understanding of the FIFO concept through an explicit spatial representation.

Table 4. Functional Testing Results for the Queue Module

No	Feature	Test Description	Result
1	Create Queue	Create a queue with a specific capacity	Successful
2	Enqueue (Push)	Add an element to the end of the queue	Successful
3	Dequeue (Pop)	Removes an element from the front of the queue	Successful
4	Display Queue	Visually displays an element	Successful
5	Front & Rear Indicator	Marking front and rear positions	Successful
6	Priority Dequeue	Removes elements based on priority	Successful
7	Auto Update State	Update the view after the operation	Successful
8	Priority Index	Mark an element as a priority	Successful
9	Clear Queue	Remove all elements from the queue	Successful

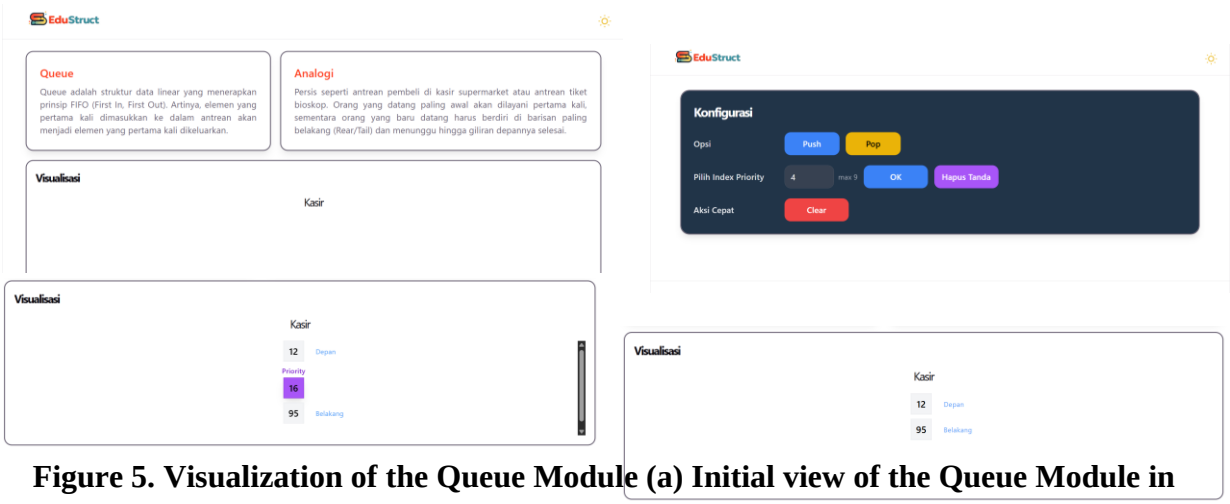


Figure 5. Visualization of the Queue Module (a) Initial view of the Queue Module in EduStruct (b) Pushing a value into the queue (c) Result after pushing a value into the queue (d) Result after popping a value from the queue

Results of the EduStruct Stack Module Implementation

The Stack Module represents a linear data structure based on the Last In First Out (LIFO) principle. The implemented operations include push, pop, and clear. The stack is visualized as a vertical structure that illustrates the concept of a stack. The last element inserted will always be at the top (*top element*).

The implementation results show that:

1 The push operation always adds an element to the top of the stack,
2 The pop operation removes the element at the top,

3 and changes in the structure are visualized through state transition animations.
This approach helps users understand the LIFO concept more concretely through visual representation.

.

Table 5. Results of the Stack Module Functional Testing

No	Feature	Test Description	Result
1	Create Stack	Create a stack with a specific capacity	Success
2	Push	Add an element to the top of the stack	Success
3	Pop	Remove an element from the top of the stack	Success
4	Display Stack	Visually display elements	Success
5	Top Indicator	Marking the top position	Success
6	Auto Update State	Update the display after the operation	Success
7	Push Animation	Element addition animation	Success
8	Animation Pop	Element removal animation	Success
9	Clear Stack	Clear all stack elements	Success

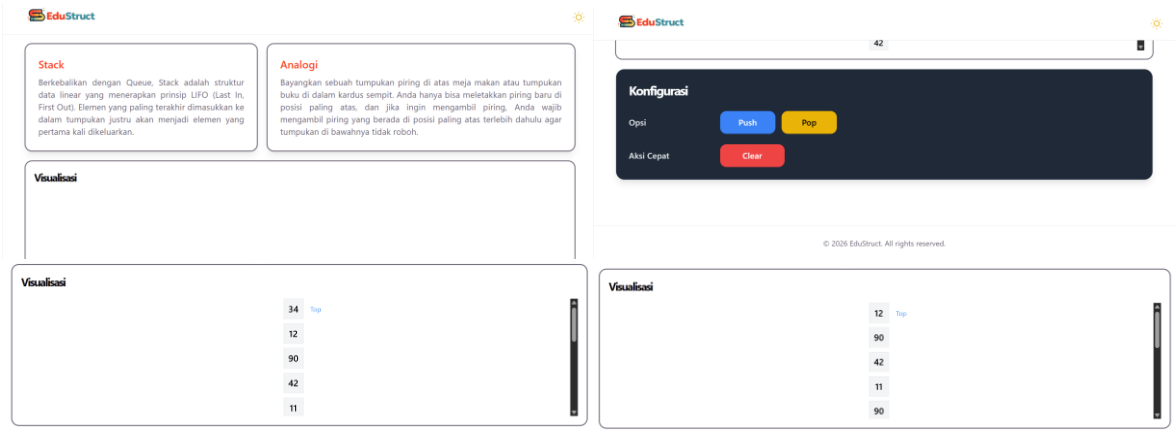


Figure 6. Visualization of the Stack Module (a) Initial view of the Stack Module in EduStruct (b) Pushing a value onto the stack (c) Result after pushing a value onto the stack (d) Result after popping a value from the stack

Interview Results Based on a Short Questionnaire

Before using EduStruct, the average scores for ease of understanding how data structures work and ease of visualizing data changes were both 4.00 (high category), and 8 out of 10 respondents agreed. However, 7 respondents cited topics they found difficult: arrays (2),

sorting (2), trees and graphs (1), linked lists (1), and priority queues (1). Three respondents reported no difficulties.

After using EduStruct, both scores rose to 4.50 (very high), and 9 out of 10 respondents agreed that EduStruct aided their understanding and that the step-by-step visualization helped them visualize data changes.

Table 6. Comparison of Pre- and Post-Scores

Aspect	Before	After	Difference	Increased / unchanged / decreased
Understanding how data structures work	4.00	4.50	+0.50	6 / 3 / 1
Visualizing Data Changes	4.00	4.50	+0.50	2/6/2

DISCUSSION

The implementation results show that the state-driven rendering visualization approach is effective in transforming abstract data structure concepts into more concrete and interactive representations. In EduStruct, every operation, such as push, pop, and enqueue is treated as a state change, and the display is updated based on that state. Students can understand data changes step by step without having to visualize them on their own.

Compared to previous systems, such as VisuAlgo, EduStruct differs in its interface design. VisuAlgo is a platform that covers many data structures and algorithms, so its interface is designed to serve a broad scope. In contrast, EduStruct focuses its interface on the list of algorithms covered in the course, including arrays, stacks, queues, and sorting so that the interface can be made simpler and more targeted for students who are just learning these concepts. The EduStruct interface displays a single main visualization area, an operation control panel, and step-by-step instructions all on one page, allowing students to follow data changes without navigating through menus. Each operation is also accompanied by text explanations for each step and indicators showing which elements are being processed, which helps students understand the relationship between the operations performed and changes in the data structure. With a state-driven rendering approach, the display always reflects the current state of the data, ensuring that the visualization students see is consistent with the results of the operations. This design is expected to reduce the cognitive load on students as they grasp these abstract concepts.

A questionnaire administered to 10 respondents showed an average increase in perceived understanding from 4.00 to 4.50 after using EduStruct, with high ratings for appearance (4.10) and ease of use (4.00). However, this increase is not yet statistically significant and is merely indicative due to the small sample size and already high baseline scores. Respondent feedback suggested the addition of introductory instructions as well as more concise and visual text presentation.

Thus, this study demonstrates that state-driven rendering can serve as the foundation for a web-based, interactive data structure learning media architecture that is easily extensible and supports the understanding of abstract concepts. By using React.js as a component-based framework, every state change can be immediately reflected in the user interface through the virtual DOM mechanism. This provides advantages in terms of responsiveness and consistency between data and visual display. These findings are consistent with previous studies indicating that web-based algorithm visualizations can enhance conceptual understanding by reducing cognitive load during the learning process. However, the system still has limitations in:

1. the scope of data structures, which is still limited to the basic level,
2. the lack of quantitative evaluation of improvements in user understanding,
3. and the lack of large-scale user testing.

Nevertheless, EduStruct shows potential as a visual-based learning medium that can bridge the gap between theoretical concepts and algorithmic implementation.

CONCLUSION

This study successfully developed EduStruct, a web-based platform used to interactively visualize data structures. This system allows users to understand basic data structure operations, such as arrays, through dynamic and responsive live simulations.
The implementation results show that all four modules in EduStruct function properly, where all operations, such as creating indexes, adding, deleting, and updating elements, as well as sorting elements from smallest to largest or vice versa can be visualized in real time. This demonstrates that a web-based visual approach can help bridge abstract data structure concepts, making them easier to understand.
Overall, EduStruct contributes to the field of data structure education by providing an interactive medium that is more intuitive than traditional text-based learning methods.

LIMITATIONS

This study has several limitations that must be considered when interpreting the results. These limitations arise due to the system's development scope, which is still at the prototype stage.

First, EduStruct is still limited to several basic data structures, namely arrays, stacks, queues, and sorting. Consequently, the system cannot yet represent the full complexity of advanced data structures such as trees, graphs, or hash tables. This limitation means that the available learning content does not yet fully cover advanced data structure material.
Second, the system is not yet equipped with quantitative learning effectiveness assessments, such as pre-tests and post-tests. Consequently, the impact of using EduStruct on improving user understanding cannot yet be objectively measured, so evaluation remains limited to the system's functional aspects and visualization.

Third, system testing remains focused on internal functions (functional testing) and has not yet extensively involved external users. This situation means that validation of the user experience and learning effectiveness is still limited to development scenarios.

Nevertheless, these limitations do not diminish the primary value of this research, namely as an interactive visualization tool to aid in the understanding of basic data structure concepts.

REFERENCES

Ardiyanto, R., & Ardhianto, E. (2024). Analisa performasi metode rendering website: Client side, server side, dan incremental static regeneration. *Computer Science (CO-SCIENCE)*, *4*(1), 19–27.

Bismoputro, I., Huda, F. A., & Brata, A. H. (2024). Pengembangan Single Page Application Berbasis Reactjs Untuk Usaha Percetakan Online (Studi Kasus: Global Grafika). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *8*(7). https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13985

Bawamenewi, D., Tanoto, C., Banjarnahor, J., Nababan, N. K., Purba, W., & Aisyah, S. (2022). Implementasi metode selection sort dalam sistem repository skripsi. *Jurnal Sistem Informasi dan Ilmu Komputer Prima*, *5*(2).

Dinanti, A., Gultom, J. R., Muchtar, I. Z., & Hutabarat, A. B. (2025). Perancangan dan implementasi website sistem antrian bank berbasis queue untuk efisiensi pelayanan nasabah. *Teknik: Jurnal Ilmu Teknik dan Informatika*, *5*(2), 93–107. https://doi.org/10.51903/teknik.v5i2.976

Ginting, S. H. N., Effendi, H., Kumar, S., Marsisno, W., Sitanggang, Y. R. U., Anwar, K., Santiari, N. P. L., Setyowibowo, S., Sigar, T. R., Atho'illah, I., Setyantoro, D., & Smrti, N. N. E. (2023). *Pengantar struktur data*. PT Mifandi Mandiri Digital.

Gustina, G., Ahadi, A. H., Aminuddin, F. H., & Syawal, M. F. (2025). Analisis kinerja algoritma Bubble Sort, Selection Sort, dan Insertion Sort menggunakan Python pada data mahasiswa. *Jurnal Inovasi Pendidikan dan Teknologi Informasi (JIPTI)*, *6*(2), 408–419. https://doi.org/10.52060/jipti.v6i2.3792

Iskandar, J., Suhendar, H. ., & Pamungkas, B. D. . (2023). Analisis Strategi Algoritma Sorting Menggunakan Metode Komparatif pada Bahasa Pemrograman Java dengan Python . *G-Tech: Jurnal Teknologi Terapan*, *8*(1), 104–113. https://doi.org/10.33379/gtech.v8i1.3556

Jafnihirda, L., Suparmi, Ambiyar, Rizal, F., & Pratiwi, K. E. (2023). Efektivitas perancangan media pembelajaran interaktif e-modul. *INNOVATIVE: Journal of Social Science Research*, *3*(1), 227–239. https://j-innovative.org/index.php/Innovative

Maspupah, A. (2024). Literature review: Advantages and disadvantages of black box and white box testing methods. *Jurnal Techno Nusa Mandiri*, *21*(2), 151-162. https://doi.org/10.33480/techno.v21i2.5776

Maravas, A., & Pantouvakis, J.-P. (2025). Schedula anima: Dynamic visualization of Gantt charts and resource usage graphs in project scheduling. *Buildings*, *15*(3), 393. https://doi.org/10.3390/buildings15030393

Nasir, J., Melladia, Saputra, R., Yusliyenni, Safaria, S., Putra, D. E., Efendi, G., Zahmi, A., & Setiawan, Y. L. (2024). *Algoritma dan Struktur Data dengan Pemrograman Pascal Dan Phyton* (R. Mulya, Ed.). CV. Gita Lentera.

Nasution, & Iswari, L. (2021). Penerapan React JS pada pengembangan front-end aplikasi startup Ubaform. *AUTOMATA*, *2*(2).

Putri, G. M., Pradja, K. A. D., Azizi, M. B. M., Nurwahid, P., Perdana, A. S., & Munawir. (2024). Implementasi stack dan array pada pengurutan lagu dengan metode selection sort. *Jurnal Teknologi dan Sistem Informasi Bisnis (JTEKSIS)*, *6*(2), 286–296. http://jurnal.unidha.ac.id/index.php/jteksis

Santoso, M. F. (2021). Teknik single page application (SPA) layout web dengan menggunakan React JS dan Bootstrap. *Jurnal Khatulistiwa Informatika*, *9*(2), 107–114.

Sholeh, M., & Rachmawati, R. Y. (2022). Pembelajaran struktur data berbasis visualisasi dengan menggunakan Visualgo.Net. *Jurnal Pemberdayaan Ekonomi*, *1*(2), 53–60. https://doi.org/10.35912/JPE.v1i2.753

Simbolon, A. P. H., Sagala, K. F., Tanjung, M. R. A., Zai, T. S. W., & Ramadhani, F. (2024). Implementasi sistem pengelolaan pesanan menu restoran berbasis stack dan queue. *Bit-Tech (Binary Digital - Technology)*, *7*(2). https://doi.org/10.32877/bt.v7i2.1867

Soetanto, H. (2022). *Struktur data dasar-dasar pemrograman*. Penerbit Universitas Budi Luhur.

Suryani, Uddin, B., & Hardianti. (2026). Visualisasi volume benda putar berbasis web untuk pendukung mata kuliah kalkulus. *SAP (Susunan Artikel Pendidikan)*, *10*(3), 402. http://dx.doi.org/10.23960/jitet.v12i2.4116

Tanjung, A. H., Marlina, L., & Yazid, A. (2025). Pembuatan aplikasi pembelajaran pengembangan website HTML dan CSS dengan animasi interaktif menggunakan Macromedia Flash 8. *Jurnal Minfo Polgan*, *13*(2), 2557. https://doi.org/10.33395/jmp.v13i2.14544

Triyono, Adithya, M. F., Putra, A. F. A., Husri, M. A. D., & Suharsono. (2024). Visualisasi Algoritma Sorting Menggunakan Django pada Data Aset Perusahaan. *Prosiding Seminar Nasional Sains dan Teknologi*. https://conference.ut.ac.id/index.php/saintek/article/view/2632

Uminingsih, Ichsanudin, M. N., Yusuf, M., & Suraya. (2022). Pengujian fungsional perangkat lunak sistem informasi perpustakaan dengan metode black box testing bagi pemula. *STORAGE – Jurnal Ilmiah Teknik dan Ilmu Komputer*, *1*(2), 1–8. https://doi.org/10.551231

Winarsih, S. M. S., & Wahono, A. W. (2022). Implementasi Dan Pengujian Struktur Data Berbasis Acuan Untuk Program Aplikasi Mengungkap Kepribadian Berdasarkan Tanggal Lahir Dan Nama. *Jurnal TIKomSiN*, *10*(2), 11. https://doi.org/10.30646/tikomsin.v10i2.631