SORTING VISUALIZATION

PROJECT REPORT

DANH SÁCH THÀNH VIÊN

Nguyễn Đình Dương	20225966	duong.nd225966@sis.hust.edu.vn
Nguyễn Mỹ Duyên	20225967	duyen.nm225967@sis.hust.edu.vn
Hồ Bảo Thư	20226003	thu.hb226003@sis.hust.edu.vn
Hà Việt Khánh	20225979	khanh.hv225979@sis.hust.edu.vn
Nguyễn Trọng Minh Phương	20225992	phuong.ntm225992@sis.hust.edu.vn
Nguyễn Lan Nhi	20225991	duong.nd225966@sis.hust.edu.vn

Giáo viên: Trần Thế Hùng

Hà Nội — 2024

MỤC LỤC

ACKNOWLEDGEMENT

We, the members of team, come together to express our deepest gratitude for the invaluable guidance and support received during the completion of our project on Sorting Visualization.

First and foremost, we would like to extend our heartfelt thanks to Ph.D. Trần Thế Hùng, our esteemed teacher, for his profound advice and unwavering encouragement. His expertise was instrumental in steering us in the right direction and ensuring the success of our project. His insightful feedback and constructive criticism have been invaluable in refining our work and enhancing our understanding of the subject matter.

We are also profoundly thankful for the opportunity to collaborate on this project as a cohesive team. Throughout this journey, we have learned and grown together, overcoming challenges and celebrating achievements. This project has not only broadened our technical skills but also strengthened our ability to work collaboratively and effectively communicate complex ideas.

Additionally, we would like to acknowledge the support and assistance provided by our classmates. Their willingness to share knowledge and help answer our queries played a crucial role in the development of our report. Their contributions have been instrumental in addressing some of the complex questions and challenges we encountered along the way.

In conclusion, we are sincerely grateful for the collective effort and support from everyone involved in this project. The knowledge and experiences gained through this endeavor will undoubtedly contribute to our future academic and professional pursuits.

SUMMARY

Our project is centered around the visualization of seven fundamental sorting algorithms: merge sort, counting sort, bubble sort, quick sort, selection sort, radix sort and shell sort, using bar charts to enhance understanding. The primary objectives are to meet our academic requirements and develop a practical and interactive Java Swing application. This application is designed to be a valuable resource for programmers, providing an engaging tool to visualize and comprehend these sorting algorithms effectively.

By visualizing the steps of each algorithm through dynamic and colorful bar charts, we aim to demystify the underlying processes of sorting. This approach facilitates a deeper understanding of how each algorithm manipulates data, highlighting their unique strategies and efficiencies.

This report encapsulates the key aspects of our project, offering a detailed examination of the contributions from team members, an extensive project overview, and a thorough exploration of our design methodology. The overview section delves into the project's requirements, providing a clear framework for our objectives and deliverables. It includes a use case diagram that illustrates the interaction between users and the application, demonstrating how the application meets user needs and enhances their learning experience.

The design section outlines the overall structure of the project and the relationships between different components. It provides insights into each package within the application, including classes, methods, and attributes, and explains their interconnections. This section highlights our systematic approach in developing the application, ensuring robustness and

maintainability.

In conclusion, this report showcases the Object-Oriented Programming (OOP) techniques employed throughout the project. It reflects our understanding and application of key programming concepts such as encapsulation, inheritance, and polymorphism. This comprehensive summary encapsulates the main points, objectives, and technical aspects of our endeavor, demonstrating our ability to apply theoretical knowledge to practical scenarios effectively. We believe that this project not only meets the academic requirements but also significantly contributes to our professional development as software developers.

1 INTRODUCTION

In the field of Computer Science, sorting is a fundamental concept essential for programmers. It involves organizing data sets that share similar properties, with various sorting algorithms developed globally. These algorithms are crucial for optimizing computational tasks. However, understanding them can be challenging, prompting the creation of a Java application as a practical solution.

Our application visualizes six specific sorting algorithms: merge sort, counting sort, bubble sort, quick sort, selection sort, and shell sort. Each algorithm has distinct characteristics, making their visualization beneficial for educational purposes. The primary objective is to enhance accessibility and engagement in understanding sorting algorithms, aligning with our class mini-project requirements. By transforming abstract concepts into interactive visual representations, we aim to facilitate a deeper comprehension of the sorting processes.

This report meticulously examines every aspect of our project, including team assignments, project descriptions, and design elements. The assignment segment serves as a grading guide, delineating individual contributions and ensuring transparency and accountability in the collaborative effort.

The project description details the application's features and requirements, including a use case diagram that illustrates user interactions. This diagram is crucial for understanding how different components work together to provide a seamless user experience.

The design section offers a comprehensive perspective through a general class diagram and scrutinizes each package within the application. It presents methods, attributes, and showcases the application of Object-Oriented Programming (OOP) techniques. These techniques, including encapsulation, inheritance, and polymorphism, are fundamental to creating a modular and maintainable codebase. This section provides insights into the architectural decisions that ensure the application's robustness and scalability.

In summary, this report documents the technical and collaborative aspects of our project, reflecting our commitment to enhancing educational tools through innovative solutions. The detailed examination of sorting algorithms and their visualization underscores the importance of practical applications in learning and demonstrates our proficiency in applying theoretical knowledge to real-world problems.

2 ASSIGNMENT OF MEMBER

1. Nguyễn Đình Dương - 20225966 (Leader)

- Package Visualizer/graphicsElements/animation/sorter/sorter.java
- Project manager

2. Nguyễn Lan Nhi - 20225991

- Package Visualizer/screen/menu/menuHelping.java
- Package Visualizer/screen/menu/menuSortingBasic.java
- Package Visualizer/screen/mainMenu.java
- Package Visualizer\graphicsElements\helping (D:\github\SortingDemonstrate\src\OOP\Grou
- Package Visualizer\graphicsElements\button\buttonImage (D:\github\SortingDemonstrate\sr
- Slide Presentation

3. Nguyễn Mỹ Duyên - 20225967

- UseCase Diagram
- Class Diagram
- Package Visualizer/screen/menu/ChatBox.java
- Package Visualizer/screen/menu/VirtualAssisstant.java
- Slide Presentation

4. Hồ Bảo Thư - 20226003

- Package Visualizer/graphicsElements/background
- Package Visualizer/graphicsElements/bars/bars.java
- Package Visualizer/graphicsElements/canvas/myCanvas.java
- Package Visualizer/graphicsElements/color/colorConcept.java
- Package Visualizer/graphicsElements/myFormatter/myFormatter.java
- Slide Presentation

5. Hà Việt Khánh - 20225979

- Package Visualizer/sorting/RadixSort.java
- Package Visualizer/sorting/BubbleSort.java
- Package Visualizer/sorting/CountingSort.java
- Package Visualizer/sorting/QuickSort.java

6. Nguyễn Trọng Minh Phương - 20225992

- Package Visualizer/sorting/Sort.java
- Package Visualizer/sorting/MergeSort.java
- Package Visualizer/sorting/SelectionSort.java
- Package Visualizer/sorting/ShellSort.java

3 PROJECT DESCRIPTION

3.1 Project Overview

The mission of our project is to build an application that visualizes six sorting algorithms, namely Merge Sort, Counting Sort, Radix Sort, Selection Sort, Quicksort, and Bubble Sort. The primary purpose of this visualization is to help users gain a deeper understanding of how these algorithms operate.

3.2 Application Restrictions

To ensure the visualization tool is both effective and efficient, we have implemented the following restrictions:

- **Element Constraints:** Only non-negative (>0) numbers are allowed to be an array's element.
- **Array Size Limitation:** The array size used for visualization has a maximum size of 200 elements.
- **Value Range Restrictions:** A valid array in Radix Sort, Merge Sort, Counting Sort, Selection Sort, Quicksort, and Bubble Sort only has the maximum value ranging from 30 to 350.

3.3 Project Requirement

For a better understanding and visualization, we have added some features to our application:

On the Main Menu

The main menu is meticulously designed to offer users an intuitive and efficient means of navigating the sorting visualizer application. The interface is composed of the following key components:

- **Title Label**: Strategically positioned at the apex of the interface, the title label prominently displays the application's name, *Sorting Visualizer*, rendered in a bold and large font to enhance visibility and user engagement.
- **Basic Sort Button**: Denoted as *Basic Sorting*, this button facilitates the initiation of basic sorting algorithm demonstrations. Upon activation, it transitions the user to a new frame dedicated to the selected basic sorting algorithm, effectively disposing of the current menu frame to streamline the user experience.
- **Help Button**: Marked as *Help*, this button summons a help dialog that provides users with critical information regarding sorting algorithms and the operational features of the application. This dialog is essential for assisting users in comprehending the application's functionalities and the theoretical underpinnings of the sorting algorithms presented.
- **Quit Button**: Identified as *Quit*, this button triggers a confirmation dialog to ascertain the user's intent to exit the application. Upon confirmation, the application terminates, ensuring that user actions are deliberate and reducing the risk of accidental closures.

The main menu is structured to provide a seamless and user-friendly interface, facilitating access to various functionalities of the sorting visualizer application. It ensures that users can easily navigate to different sorting demonstrations and access necessary information, thereby enhancing the overall usability and educational value of the application.

In the demonstration

- **Chatbox Button**: Positioned in the bottom-left corner of the interface, this button allows users to access an integrated chatbox. The chatbox serves as an interactive tool where users can pose questions related to sorting algorithms. This feature enhances user engagement and provides real-time assistance, fostering a deeper understanding of the algorithms being visualized.
- **Back to Menu Button**: Situated adjacent to the Chatbox Button, this button facilitates users' return to the main menu. It offers seamless navigation between the active sorting demonstration and the main application interface, thereby improving the user experience by allowing easy switching between different sections of the application.
- **Choose an Algorithm**: This feature enables users to select from the six available sorting algorithms that they wish to visualize. By providing a range of algorithm options, the application caters to diverse user interests and educational needs, allowing for a comprehensive exploration of sorting techniques.
- **Capacity**: This option allows users to define the capacity of the data set being sorted. By adjusting the number of elements, users can observe the performance and behavior of sorting algorithms under different data loads, thereby gaining insights into their efficiency and scalability.
- **FPS**: The Frames Per Second (FPS) setting provides users with the ability to modify the speed at which the sorting algorithm is displayed. This feature is crucial for educational purposes, as it enables users to slow down the visualization to better understand the sorting process or speed it up to observe the algorithm's performance at higher speeds.

3.4 Use Case Diagram and Explanation

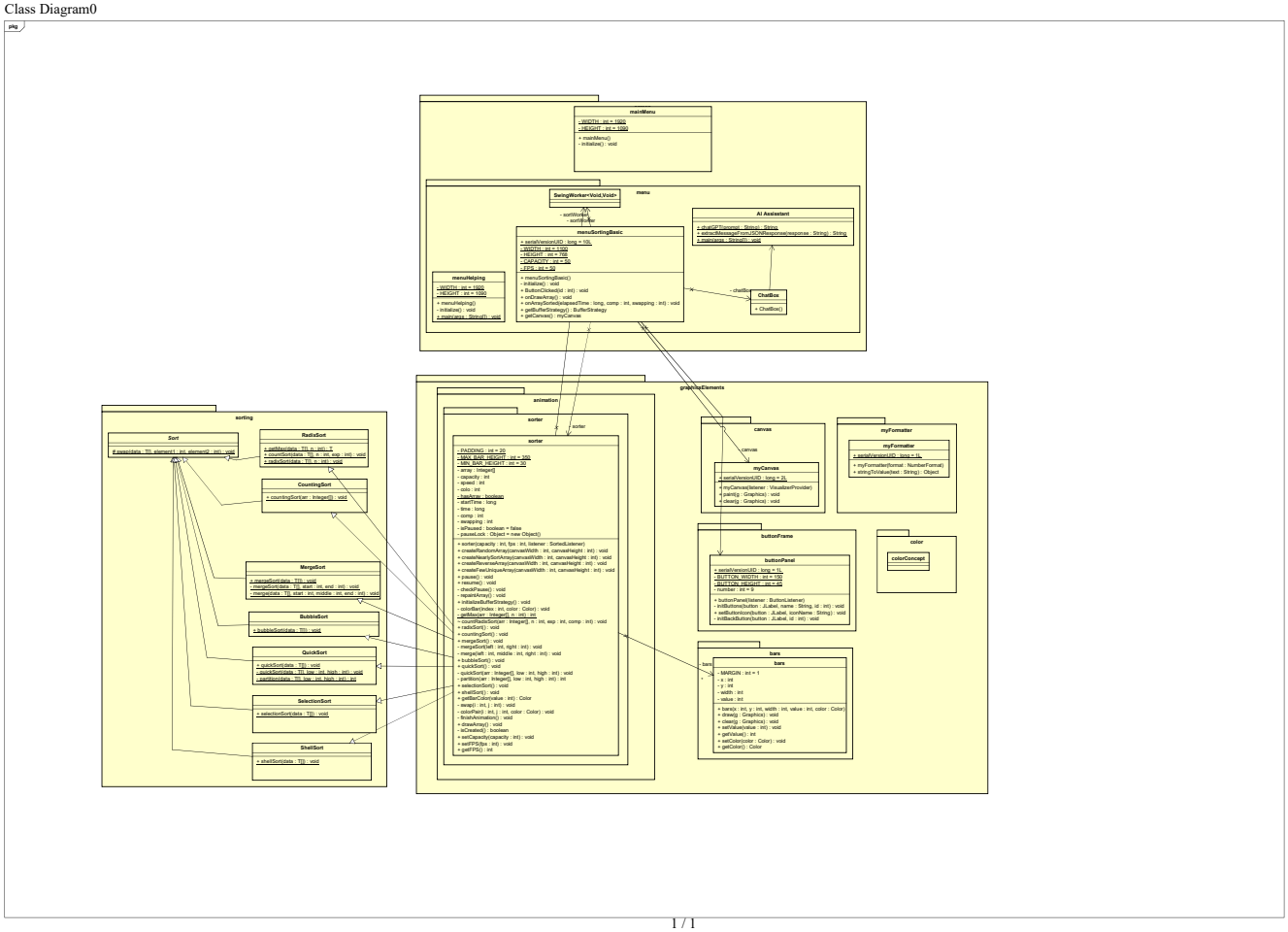
Hình 1: Use case diagram.

Based on all the requirements, we decided to develop six use cases (as shown in the figure for our application). To be more specific:

1. **Choose Sort Algorithm**: This use case allows the user to select a specific sorting algorithm to apply to an array. Users can choose from a variety of algorithms including Merge Sort, Counting Sort, Radix Sort, Quick Sort, Selection Sort, Bubble Sort, and Shell Sort. This functionality enables users to explore and compare the performance and behavior of different sorting techniques.
2. **Create Array**: This use case enables the user to create an array with different initial conditions. Users can generate arrays that are Random, Nearly Sorted, Reversed, or with Few Unique elements. This flexibility in array creation helps users to observe how various sorting algorithms handle different types of input data, providing a deeper understanding of their efficiency and adaptability.
3. **See Each Step of the Sorting Visualization**: This use case provides users with a step-by-step visualization of the sorting process. Users can follow the algorithm as it performs each operation, gaining insights into its inner workings. The visualization can be paused, resumed, and stepped through at the user's preferred speed, offering a detailed and interactive learning experience.
4. **Chat with AI Bot About Algorithm**: This use case incorporates an AI chat bot that allows users to ask questions about sorting algorithms. The AI bot provides real-time assistance and answers, enhancing the user's learning experience by offering explanations and insights into the algorithms being visualized. This interactive feature helps users to resolve doubts and gain a deeper understanding of complex concepts.
5. **Get Help**: This use case provides users with access to a help section where they can find detailed information about the application and the sorting algorithms it supports. The help section includes tutorials, FAQs, and detailed documentation that guides users through the application's features and functionalities, ensuring they can effectively utilize all available tools.
6. **Quit**: This use case allows the user to exit the application. It includes a confirmation dialog to ensure that the user intends to quit, preventing accidental closures. This feature ensures that users can gracefully exit the application when they have finished their tasks, preserving their work and settings as needed.

4 DESIGN

4.1 General Class Diagram



Hình 2: General Class Diagram

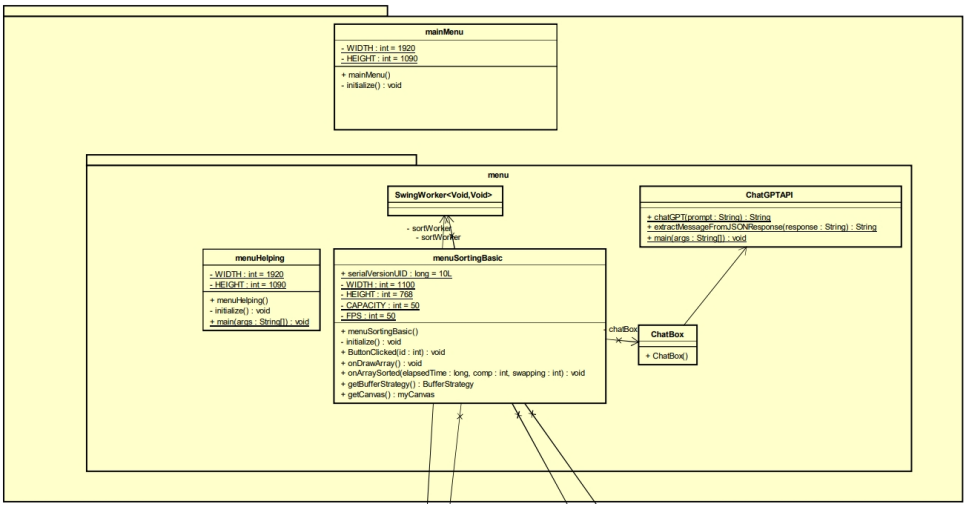
4.2 Detailed Class Diagram of Each Package

4.2.1 Package Screen

The `screen` package contains the main display screens and auxiliary frames within the menu. Its key components include:

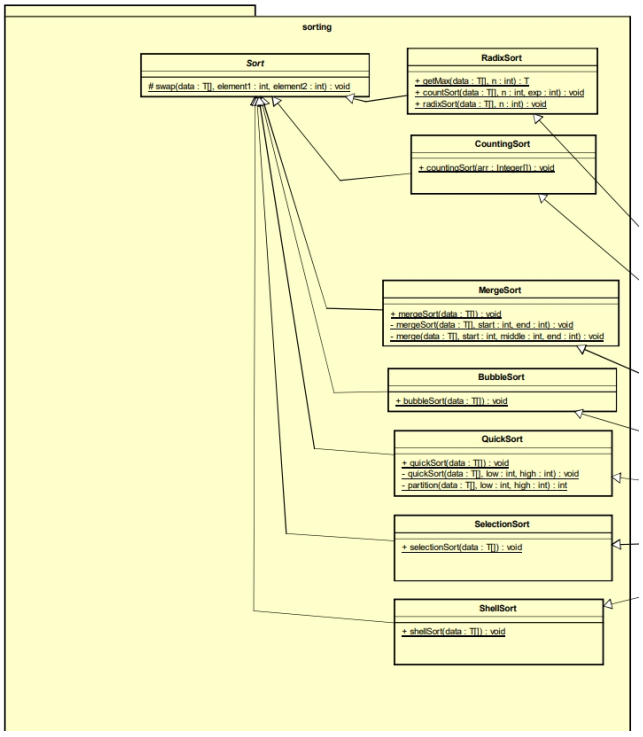
- `mainMenu.java`: The main menu of the application where users can navigate to different sections.
- `menu/`
  - `ChatBox.java`: A chat interface for user interaction.
  - `menuHelping.java`: A help menu providing guidance and instructions.
  - `menuSortingBasic.java`: The primary menu for sorting algorithms, integrating various UI elements.

– `VirtualAssisstant.java`: A virtual assistant to help users navigate and use the application.



Hình 3: Package Screen

4.2.2 Package Sorting



Hình 4: Enter Caption

The `sorting` package includes the various sorting algorithms implemented in the application. Each class represents a different sorting method:

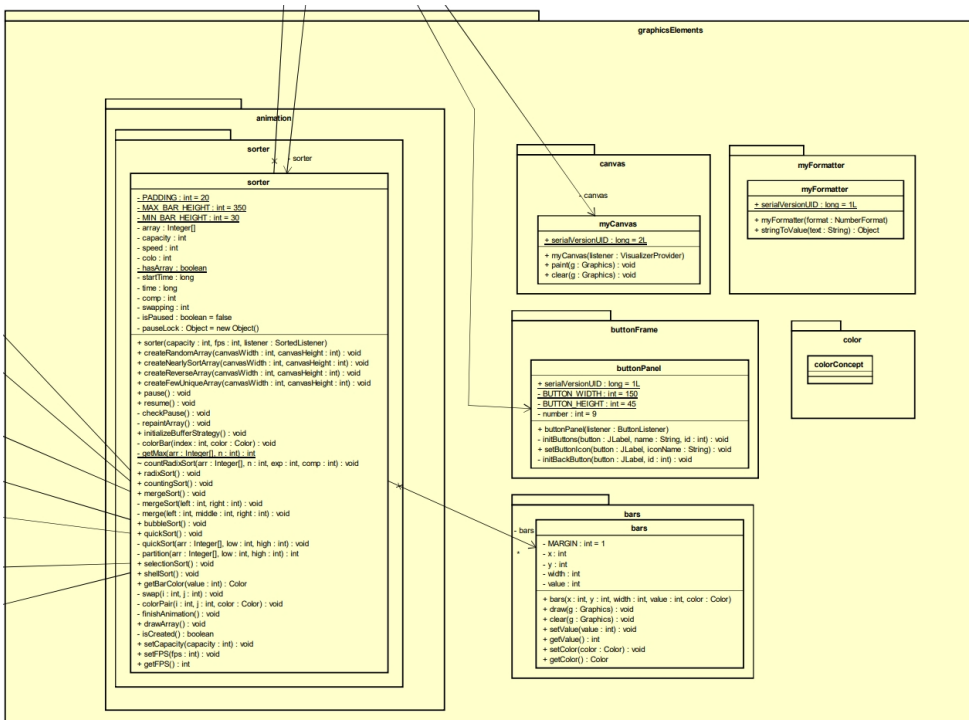
- `BubbleSort.java`: Implements the bubble sort algorithm.
- `CountingSort.java`: Implements the counting sort algorithm.
- `MergeSort.java`: Implements the merge sort algorithm.
- `QuickSort.java`: Implements the quicksort algorithm.
- `RadixSort.java`: Implements the radix sort algorithm.

- `SelectionSort.java`: Implements the selection sort algorithm.
- `ShellSort.java`: Implements the shell sort algorithm.
- `Sort.java`: A base class providing common functionalities for sorting algorithms.

4.2.3 Package GraphicsElements

The `graphicsElements` package encompasses the functions related to graphical elements and visualizations in the application. Its key components include:

- `animation/sort/sorter.java`: Handles animation tasks for sorting, displayed on the canvas frame in `menuSortingBasic.java`.
- `background/`: Contains background images used in the application.
- `bars/bars.java`: Constructs the visual representation of an array column on the canvas frame in `menuSortingBasic.java`.
- `button/buttonFrame/buttonPanel.java`: Designs the visual appearance and positioning of function buttons in `menuSortingBasic.java`.
- `canvas/myCanvas.java`: Builds the canvas to be implemented in `menuSortingBasic.java`.
- `color/colorConcept.java`: Contains color schemes used in the program.
- `helping/`: Provides additional helper functionalities.
- `myFormatter/myFormatter.java`: Converts string values to integers.
- `env.java`: Contains environment variables for the application.



Hình 5: Enter Caption

CONCLUSION

The Java Swing-based application has successfully fulfilled all the project requirements outlined within the course. Extensive testing has confirmed its functionality and ensured its runnability. The visualization components, specifically for Merge Sort, Counting Sort, Radix

Sort, Quick Sort, Selection Sort, Bubble Sort, and Shell Sort, have been implemented with a high degree of efficacy. Visualization, being a multifaceted medium of expression, can adopt various styles to effectively convey complex concepts. This project has achieved a unique representation of sorting algorithms through a Java application, providing insights that extend beyond mere lines of code and syntax.

The application's architecture is meticulously organized into five principal packages, reflecting a coherent and systematic design. Throughout the development process, we have employed several Object-Oriented Programming (OOP) paradigms, including Inheritance, Polymorphism, Aggregation, and Composition, within the bounds of our expertise. Despite the project's success, there remain numerous opportunities for further enhancement. Potential areas for improvement include refining the User Interface (UI) design, enhancing accessibility, optimizing application logic and performance, and integrating additional tools.

Visualization serves as a critical conduit for communication between algorithms and programmers. Our overarching objective moving forward is to further develop the project to attract a broader audience and foster a deeper understanding of these abstract algorithmic concepts. By doing so, we aim to elevate the educational value and usability of the application, ensuring it remains a valuable resource for both students and professionals in the field of computer science.