**INTERNATIONAL UNIVERSITY**

**VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY**

**School of Computer Science and Engineering**

**-----\*\*\*-----**

**PROJECT REPORT**

**SORTING VISUALIZER**

**ALGORITHMS AND DATA STRUCTURES (IT013IU)**

**Course by Dr. Tran Thanh Tung**

**Student’s ID**

ITITWE20024

ITDSIU21055

ITDSIU21057

ITITDK21071

**TEAM MEMBERS:**

Phan Dinh Thanh Hoan

Hoang Tuan Kiet

Nguyen Hai Ngoc

Tran Nguyen Trung Quan

**TABLE OF CONTENTS**

[**LIST OF FIGURES** 2](#_Toc135687794)

[**CONTRIBUTION TABLE** 3](#_Toc135687795)

[**ABSTRACT** 4](#_Toc135687796)

[**CHAPTER 1. INTRODUCTION** 5](#_Toc135687797)

[1. Objectives 5](#_Toc135687798)

[2. The tools used 5](#_Toc135687799)

[**CHAPTER 2. METHODOLOGY** 7](#_Toc135687800)

[1. Rules 7](#_Toc135687801)

[2. Design 7](#_Toc135687802)

[2.1. UI/UX 7](#_Toc135687803)

[2.2. Project algorithms 8](#_Toc135687806)

[3. UML Diagram 10](#_Toc135687807)

[**CHAPTER 3. DEMO – RESULT** 12](#_Toc135687808)

[**CHAPTER 4. CONCLUSION AND FUTURE WORKS** 13](#_Toc135687809)

[1. Conclusion 13](#_Toc135687810)

[2. Future work 13](#_Toc135687811)

[3. Acknowledgement 13](#_Toc135687812)

[**CHAPTER 5. REFERENCES** 14](#_Toc135687813)

# **LIST OF FIGURES**

[Figure 1.1. Sorting Visualizer 5](#_Toc135687842)

[Figure 1.2. A class in VS Code 6](#_Toc135687843)

[Figure 1.3. GitHub working environment 6](#_Toc135687844)

[Figure 2.1. Types of sorting algorithms 7](#_Toc135687845)

[Figure 2.2. Speed selection 8](#_Toc135687846)

[Figure 2.3. Reset button 8](#_Toc135687847)

[Figure 2.4. Sort button 8](#_Toc135687848)

[Figure 2.5. Visualization area 8](#_Toc135687849)

[Figure 2.6. Project structure 9](#_Toc135687850)

[Figure 2.7. UML of the whole project 10](#_Toc135687851)

[Figure 2.8. UML of Components related classes 11](#_Toc135687852)

[Figure 3.1. Generating the project 12](#_Toc135687853)

[Figure 3.2. A sorted array 12](#_Toc135687854)

# **CONTRIBUTION TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Full Name | Student’s ID | Contribution |
| 1 | Phan Dinh Thanh Hoan | ITITWE20024 |  |
| 2 | Hoang Tuan Kiet | ITDSIU21055 |  |
| 3 | Nguyen Hai Ngoc | ITDSIU21057 |  |
| 4 | Tran Nguyen Trung Quan | ITITDK21071 |  |

# **ABSTRACT**

Sorting Visualizer is an interactive online platform that visualizes sorting algorithms. The team’s purpose is to provide users with a webpage to comprehend data structures and algorithms, especially sorting algorithms.

In this undertaking, the team created the platform to visualize algorithms by allowing users to create arrays, sorting in various sorting algorithms, and exploring each algorithm’s complexity. Each array is randomly generated and then visualized based on specific rules of the algorithm. The visualizer has been designed to be user-friendly, by arranging the theme with a simple and effective graphical user interface.

*Keywords: sorting, visualizer, algorithms, data structures, arrays, complexity, graphical user interface.*

# **INTRODUCTION**

## Objectives

The project's goal is to create a visualization tool for sorting algorithms using React JS. This can also demonstrate some basic features of the sorting algorithms in DSA. To further enhance the tool, commitments from others may also be acceptable.

To be short, the project aims to:

* Visualizing sorting algorithms in action.
* Contributing to developing people’s understanding of DSA.
* Go through the process of web-tool programming and code refining.
* Evaluate the ability to build more features on top of the basic code.

A picture containing text, design

Description automatically generated

Figure 1.1. Sorting Visualizer

## The tools used

* IDE for programming and debugging: Visual Studio Code
* Mean of code version management: GitHub.

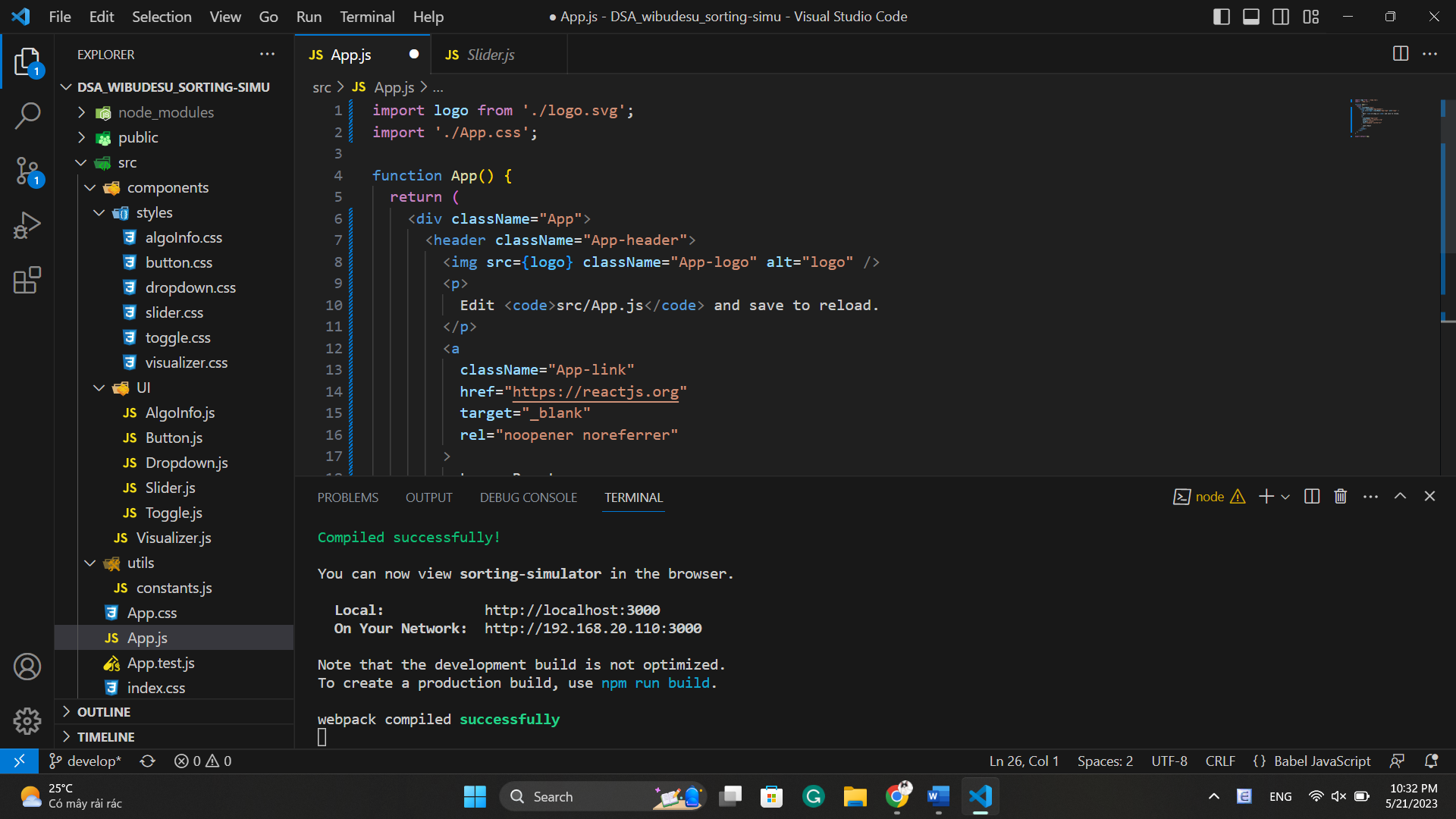


Figure 1.2. A class in VS Code

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1.3. GitHub working environment

# **METHODOLOGY**

## Rules

The team tried to build a web screen with simple functions that users can easily adapt, here is the summary of the main points:

* Sorting selection: There are 6 types of sorting algorithms provided, including Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort. Users can select one type of the desired sorting to do visualization.
* Speed selection: Provided 3 different speeds, including Slow, Medium, and Fast, which are in ascending order of speed values.
* Reset button: provided for generating a new array for sorting, and clearing all of the previous actions.
* Sort button: provided for starting to apply the sorting algorithms to the array, also doing visualization corresponding to each sort.
* Visualization area and sorting information: provided some bar to visualize the algorithms and an area to show the algorithm’s information, including its time complexity.

## Design

### UI/UX

* Sorting selection

A screenshot of a computer

Description automatically generated

Figure 2.1. Types of sorting algorithms

* Speed selection

A picture containing text, screenshot, circle, font

Description automatically generated

Figure 2.2. Speed selection

* Reset button

A yellow rectangle with white text

Description automatically generated with medium confidence

Figure 2.3. Reset button

* Sort button

A pink rectangular object with white text

Description automatically generated with low confidence

Figure 2.4. Sort button

* Visualization area and sorting information

*A picture containing candle

Description automatically generated*

Figure 2.5. Visualization area



### Project algorithms

Our final project structure is below:

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2.6. Project structure

Initially, the project has been separated into 3 parts: **components** package, **utils** package, and groups of testing applications class for each type of function class. The **components** package is the main class that includes almost our project, it contains **styles** and **UI**, which will be described in detail below:

* 1. Style: Including classes in CSS type, styling, and aligning the tool’s layout.
* AlgoInfo
* Button
* Dropdown
* Slider
* Toggle
* Visualizer
  1. *UI:* Including classes in javascript (js) type:
* AlgoInfo: visualization of the complexity of each sorting algorithm.
* Button: creating interactive Sort and Reset button.
* Dropdown: creating interactive sorting selection.
* Slider: creating interactive speed selection.
* Toggle
  1. *Visualizer.js:* Including visualization of the algorithm’s information, sorting handling (randomly generating array, time limitation), and the basic algorithm rules of each sort in Javascript.

About the app classes, they will respectively test the class in the type of css, javascript, and reactJS.

## UML Diagram

To better visualize the structure and algorithms, we included the UML diagrams for the whole project and each group mentioned.

* For the whole project

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2.7. UML of the whole project

* For the scope: components (without styles package)

A screenshot of a computer

Description automatically generated

Figure 2.8. UML of Components related classes

# **DEMO – RESULT**

To install and run the project, users should run the code below in the terminal:

*> npm install*

*> npm start*

After generating these lines, the screen will be switched to the users’ default web app, and the page will be opened similarly to the figure below:

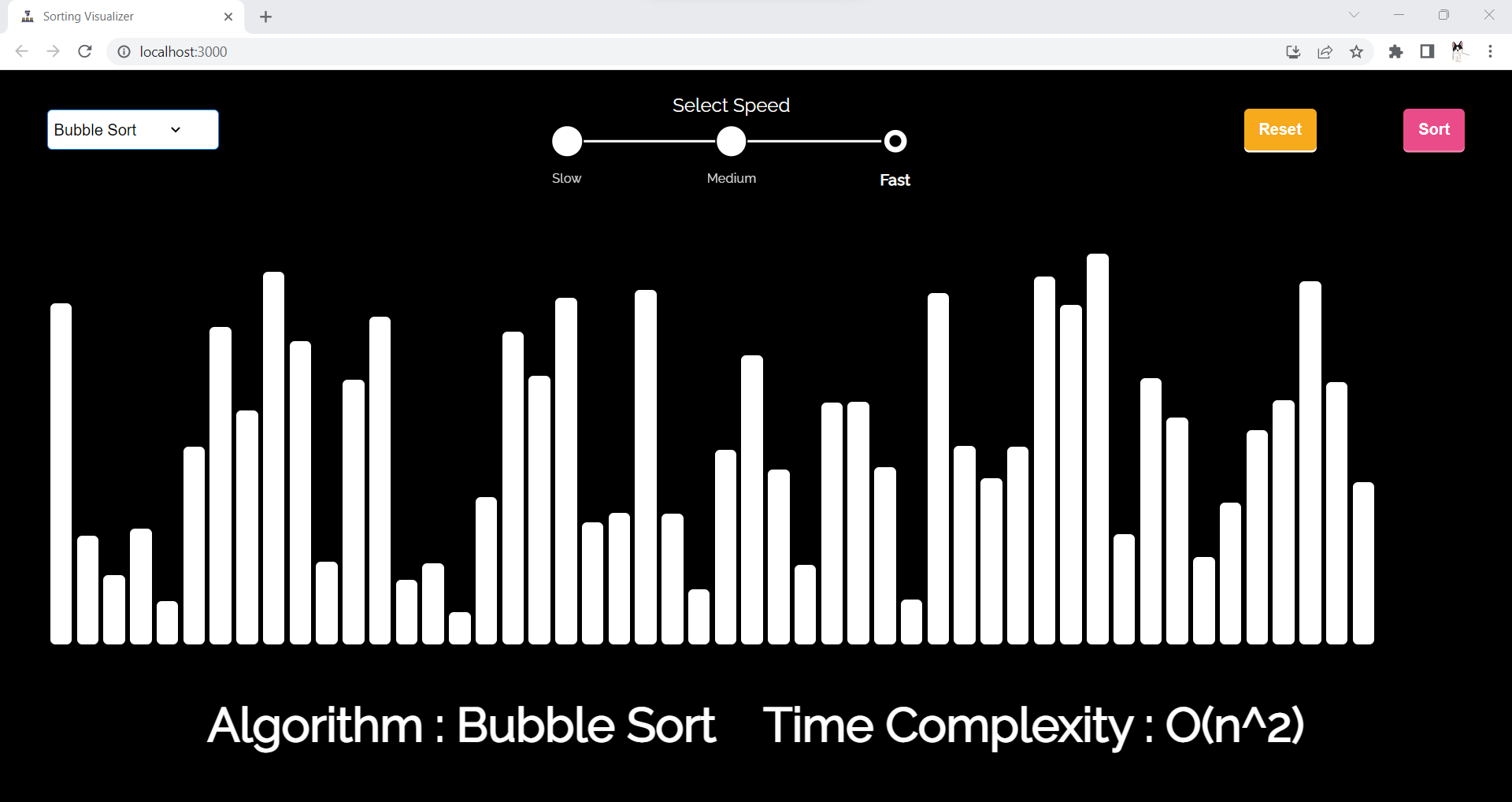


Figure 3.1. Generating the project

An example after generating bubble sort:

A picture containing screenshot, text, line, colorfulness

Description automatically generated

Figure 3.2. A sorted array

# **CONCLUSION AND FUTURE WORKS**

## Conclusion

The tool of sorting visualization has not yet ended. After running the project, the team has accomplished a deeper understanding of DSA applications, especially in sorting algorithms, in the variety of programming languages used to build a web tool such as Sorting Visualizer. Through the project, the team hopes that this tool will be an accurate technique for users working in the field of DSA, mostly in sorting algorithms.

## Future work

The team hopes to develop a visualizer of more algorithms and data structures. Shortly, the searching algorithms, as well as some basic data structures such as stack, queue, or binary tree will be included to add extra resources for users and extend our project features. According to that, a selection bar for switching to the algorithms or data structures will be applied to display the available fields. Furthermore, the display of the tool will be improved to provide users with the most interesting experience in using the tool simulator.

## Acknowledgement

We want to express our sincerest thanks to our lecturer and the people who have helped us to achieve this project's goals:

* Dr. Tran Thanh Tung
* MSc. Pham Quoc Son Lam
* MSc. Nguyen Quang Phu

# **REFERENCES**

[1]: *React – A JavaScript library for building user interfaces*. (n.d.). React. <https://legacy.reactjs.org/?fbclid=IwAR0MnlcoT7tEfEJsy4YOwZdD5Bads_tMJXNh4-6FPD2sNCPxiG036BXSiwY>

[2]: *Documentation | Framer for Developers*. (n.d.). <https://www.framer.com/motion/?fbclid=IwAR2GuUvDpP_rHt0V93Wm5qWW9Nxn-xKkj0ivxKcUCbtfF-mioR48XRJZv5w>

[3]: *Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell*. (n.d.).<https://www.bigocheatsheet.com/?fbclid=IwAR2nUR_w1K3Xrh8cyX35jVDf2Kuz97u8CWFSn1mtpn_G_e1zx8aNcC7ELiw>

[4]: *Window: localStorage property - Web APIs | MDN*. (2023, April 8).

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage?fbclid=IwAR3A-P5XsJgYd_WwIxOH3-2INPb2bqTqlHnoTkBAETAaMAGvQpezIm2Jf9>