Algorithm is an integral part of computers and programming. It provides consistency in solving similar tasks with a predictable and desirable outcome. Among algorithms, sorting algorithms are regarded as the gateway to the world of algorithms in programming. Evident by their definition, they are used to sorting data set with similar properties. There are various types of sorting algorithms with their distinct implementations and strategies. They are often not as easy to understand and absorb. A possible contributing factor can be the lack of visualization of the process and steps each one takes to accomplish the task of sorting. Thus, the objective of this thesis was to create a web application as a visualization tool for three common sorting algorithms namely Bubble sort, Merge sort, and Quick sort.

The result was a web application built using React and other modern JavaScript technologies. The application UI was able to visualize the sorting process using various colors or denote the status of the elements in the array. It includes features to select the size of the data (array of random numbers), visualization speed, sorting algorithm. Along with that, an extra feature for accessibility called color mode is also present for anyone with a medical condition that is affected by flashing colors.

The main objective of the thesis was met. However, the application can be improved further in terms of responsiveness and support mobile devices, design, and more options of sorting algorithms, etc.
Contents

List of Abbreviations

1 Introduction 1

2 Theoretical Background 2
   2.1 Algorithm 2
   2.2 Sorting Algorithm 3
   2.3 Measurement of algorithm 4
   2.4 Visualization 5

3 Objective 7

4 Research Methodology 7
   4.1 Waterfall Methodology 7
   4.2 Agile Methodology 9
   4.3 Flow chart of the application logic 10

5 Web application as a Visualization tool 11
   5.1 HTML, CSS 12
   5.2 JavaScript and TypeScript 13
   5.3 React 14
   5.4 Other tools 15
      5.4.1 Node.js 15
      5.4.2 Redux 16
      5.4.3 React Router 16
      5.4.4 Sass 17
      5.4.5 Webpack 17
      5.4.6 Babel 17
      5.4.7 GitHub and Git 18

6 Project Implementation 18
   6.1 Building the application 18
   6.2 Implementing the visualization in React 19
   6.3 State Management 23
   6.4 Project structure 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5 Hosting</td>
<td>25</td>
</tr>
<tr>
<td>Project outcome</td>
<td>26</td>
</tr>
<tr>
<td>Conclusion</td>
<td>30</td>
</tr>
<tr>
<td>References</td>
<td>31</td>
</tr>
</tbody>
</table>
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascade Style Sheet</td>
</tr>
<tr>
<td>SASS</td>
<td>Syntactically Awesome Style Sheets</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>JIT</td>
<td>Just-in-time</td>
</tr>
<tr>
<td>ECMA</td>
<td>European Computer Manufacturers Association</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>JSX</td>
<td>JavaScript XML</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>SPA</td>
<td>Single Page Application</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>HLD</td>
<td>High-Level Design</td>
</tr>
<tr>
<td>LLD</td>
<td>Low-Level Design</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>GCP</td>
<td>Google Cloud Platform</td>
</tr>
</tbody>
</table>
1 Introduction

In the present time, the use of electronic devices in our daily life has seen exponential growth both in terms of number and diversity. Smart systems are being integrated into every possible item such as home automation systems, mobile phones, gaming consoles, watches and fitness trackers, home appliances, weighing scales, vacuum cleaners, even shoes, and many others. These devices with integrated smart systems aim to be more productive, convenient, assistive, and interactive to use. With the increasing number of devices, the volume of data and information those devices produce is also growing at the same pace if not more.

Currently, data is the hottest commodity of the IT industry. All the large co-operations are vying to collect as much data as possible. While data collection is desired, the collection and storage of such data are time-consuming and costly. Yet, the benefits of having data collection outweigh them in the long run. Data are important to understand the performance, accuracy, approval, etc. of the product and it helps businesses analyze them and improve their productivity and shortcomings. Examples of data collected by companies can be information on the fitness tracker, internet browsing history, IP address, time, and frequency of using an app or device, etc. As important it is to collect and store data, it is also important to organize the data in a meaningful manner. Organization of data includes sorting of data, it provides proper structure to the data that makes it analyzable, comparable to another set of data, and accessible as fast as possible. Thus, sorting is an indispensable part of data organization.

The thesis focuses on various such sorting algorithms that can be used to sort a set of random numbers as a data set and visualize the sorting process using a web application. This thesis report comprises 8 chapters. This introduction section covers chapter one. Chapter two covers the theoretical background. Chapter three discusses the research methodology and objective of this thesis. Chapter four consists of the description of the web application as a visualization tool and the tools used in the thesis application. Chapter five focuses on the discussion of the implementation of the application. Chapter six highlights the project outcome and chapter seven focuses on the conclusion of the thesis.
2 Theoretical Background

This section introduces the theoretical backgrounds of the various aspects that are part of the thesis in detail. The topics include algorithm, sorting algorithm, visualization with their definition, history, and scope.

2.1 Algorithm

Algorithms can be defined as a set of steps to accomplish a given task. Its purpose is to make it easier to replicate doing the task again and be efficient and accurate to the same degree. The term ‘Algorithm’ is derived from the name of 9th-century Persian scientist, astronomer, and mathematician Abdullah Muhammad bin Musa al-Khwarizmi who is also regarded as ‘The father of Algebra’. [1]

Algorithms can be found everywhere in day-to-day life. The primitive form of algorithm began with calculating and at present, it is being used in complex study and operations such as artificial intelligence, computer processing, gaming, molecular biology, and various other scientific research areas. It can be as simple as steps to turn on and off a light switch to any complex mathematical or computational operations. It is used primarily for calculation, data processing, and automated reasoning. The importance of algorithms arises once quality and efficiency are key factors to be considered while accomplishing any task. The opposite of following an algorithmic process is performing the same task, each time in a unique manner. This introduces unreliability, inefficiency and increases time, energy, cost, and many other factors. Thus, the use of algorithms helps to eliminate these hurdles, brings consistency, efficiency, and quality which is visible as the result.

Algorithm is related to function and sometimes they are even used interchangeably but still, they are not the same. While an algorithm is a set of ideas or an abstract concept that elaborates on how to solve a problem, functions are the actual implementation of the algorithm which can be used to solve them. Hence, a function can be an implementation of a whole algorithm or a single or multiple steps of an algorithm.
2.2 Sorting Algorithm

Sorting is the process of arranging a set of data in sequenced increasing or descending order based on common properties among the data items. The properties could be name, price, height, weight, length, efficiency, etc. The primary purpose of sorting any data set is to increase the time efficiency of searching and retrieving the required item from it. Sorting is common in the things we interact with or use in our daily life, yet it is often unnoticed. Language Dictionary, Telephone directories, point table in any sport, posts in social media, waste sorting, etc. are a few examples of sorting in action in daily life. Sorting used in these mentioned things helps us retrieve details like a word in the dictionary, telephone number by the name, point of our favorite team, recycling, etc. faster and easier.

As described in the section above, it is desirable to use an algorithm to perform the task for efficiency, consistency, and quality of the result. Similar is true for sorting, hence, there are algorithms for sorting which are referred to as Sorting Algorithms. There are numerous sorting algorithms, each is implemented with a different approach and logic. They have varying efficiency and are better suited for cases with different data distribution. They can be classified into multiple categories based on sorting techniques, few of the techniques are listed below: [2]

- Comparison/Non-comparison-based sorting: A sorting is comparison-based sorting if the elements are compared with each other to find the sorted array otherwise, it is non-comparison-based sorting.

- In-place/Out-place sorting: A sorting is in-place sorting if it does not use extra memory to sort the array otherwise, it is out-place sorting.

- Online/Offline sorting: A sorting is online sorting if it allows new data while the sorting process is ongoing and regardless yields the result otherwise, it is offline sorting.

- Stable/Unstable sorting: A sorting is stable sorting if it does not change the order of place of an element with the same value otherwise, it is unstable sorting.
Table 1 below shows a list of different sorting algorithms. The table contains information on inventors, the techniques of sorting used in the sorting algorithms. The listed algorithms are some of the most common sorting algorithms.

<table>
<thead>
<tr>
<th>Sorting algorithm</th>
<th>Inventor</th>
<th>Comparison based</th>
<th>In-place</th>
<th>Online</th>
<th>Stable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge Sort</td>
<td>John Von Neumann</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Quick Sort</td>
<td>Sir C. A. R. Hoare</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bubble Sort</td>
<td>Iverson</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>Unknown</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Selection Sort</td>
<td>Oscar Wilde</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Heap Sort</td>
<td>J. W. J. Williams</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1. List of sorting algorithms, their inventor, and sorting techniques they use [2]

2.3 Measurement of algorithm

Algorithms are measured for their efficiency in terms of the time they take to perform a task as the input grows and the space or memory they use. There are various terms used to describe the efficiency of an algorithm namely, Big O (\(O()\)), Little O (\(o()\)), Omega (\(\Omega()\)), and Theta (\(\Theta()\)). Big O describes the upper bound including the exact bound of the complexity. Little O describes the upper bound but excludes the exact bound. Omega describes the lower bound of the complexity while theta describes the exact bound of the complexity. [3] The best case is considered to be \(O(1)\), which means the algorithm can process without any drop in time or space even when the input grows. For example, the console.log () function of JavaScript has the complexity of \(O(1)\). Its performance is independent of the size of the input. However, complex calculations such as sorting algorithms have varying complexities.

Table 2 below provides the comparison among sorting algorithms based on time and space complexity. These are algorithms implemented in the thesis application for visualization.
Table 2. The table contains the comparison of sorting algorithms based on time and space complexity [2]

The fastest sorting algorithm in terms of time complexity is Quick sort and Insertion sort is the most efficient in terms of space.

2.4 Visualization

Visualization is a visual representation of an idea, story, data, etc. On a simplified note, using a visual clue to describe a piece of information is also Visualization. The visualized information or entity doesn't need to be a real-life or physical object, it can an abstract idea such as feelings, concepts, imagination, or just a belief. Letters, numbers, statues, symbols, arts such as paintings, sketches, murals, digital graphics, movies, etc. are means of visualization that are used to convey an idea of information. Any information that is visualized pleasingly and informatively has a greater potential of communicating the core message much easier and faster.

Visualization has been a part of human history from the early prehistoric period. The oldest piece of prehistoric art ever discovered is a rock carving named ‘Bhimbetka and Daraki-Chattan Cupules’ located in central India. It is reported to date back around 290-70,000 BC. Likewise, one of the recently discovered artifacts includes the oldest known cave painting (Figure 1) discovered in Sulawesi, Indonesia which dates back at least
37,900 BC. [4] Furthermore, there are uncountable ancient monuments and archeological sites scattered throughout the globe which represent ancient civilization, historical events, culture, and human cognitive capability of a certain period. Through these artifacts, they visualize the animals, flora-faunas, environment, human feelings, and human lifestyle prevalent in the respective region during the period to which they belong. They are of great importance to study of the human history and evolution.

Hence, visualization is not a new form of conveying messages; it has existed since the dawn of human history. Along the way, it has evolved into numerous forms and means but at its core, it serves the same purpose. In the present day, there is a large quantity of information and data with a higher level of complexity. The information must be delivered as efficiently as possible with reliability and even more accuracy. It has given rise to modern visualization in the form of data visualization for research, study, business, entertainment, books, magazines, advertisements, etc. All competing to be more visually pleasing and informative than others. Thus, visualization is of great importance and will remain so, for upcoming days.

Figure 1. World’s oldest known cave painting [5]
3 Objective

The main objective of the thesis project is to create a web application as a visualization tool. A single-page web application built using modern JavaScript technology that will visualize the flow and logic of various sorting algorithms. The UI will contain options to select one of the sorting algorithms which were implemented and several items or elements in the data array, control buttons to start, pause, navigate to previous or next steps along with an option for sorting speed and color mode. The data array of the selected size will be filled in with randomly generated unique values. The data set is represented as a vertical bar with the height of their respective values. After the sorting is started, the stepwise arrangement of data in ascending order based on their value/height will be visualized in the UI.

4 Research Methodology

This section contains a description of the research methodologies followed during the implementation of the project. The project was implemented with a mixture of both Waterfall and Agile methodologies often referred to as Hybrid Development methodology. Brief descriptions of the methodologies are given below.

4.1 Waterfall Methodology

The waterfall model is the first modern approach to the (SDLC) software development life cycle model. The model describes the project development in multiple sequential phases. Each phase track progress of the project from multiple dimensions and the result of each phase act as input for the next phase. [6]

The description of the sequential phases that comprise the waterfall model are as follow:

- Requirement analysis: In the first phase of the waterfall model, all the business requirements and logic of the system are documented. The requirements are divided into two categories viz Functional requirements and Non-Functional requirements. Functional requirements define the system behavior while Non-functional requirements define how the system should behave. These requirements
are gathered through the discussion of the end-users and/or clients and finally validated for the possibility of the implementation with given time and resources. The documented requirement serves as a guideline for the next phase.

- System design: In the second phase, all the collected requirements are taken into consideration to prepare documentation for the system and software design. In this phase, there are two types of design development High-Level Design (HLD) and Low-Level Design (LLD). HLD describes the properties of every module such as description, name, outline, functionality, relationship, identification of database table, and complete architecture diagram along with technical details. And LLD is more of a description of the functional logic of the modules, database table with the properties like type and size, complete interface details, handling errors, and input/output for every module.

- Implementation: In the third phase, the coding of the application begins according to the specification of the system design. The system is built using the chosen programming language/s. The tasks are divided into multiple smaller units which are assigned to the developers. Along the way, the unit or the modules are integrated into larger functional components of the system as described in the previous phases. This is normally the longest-running phase-out of all the phases in the waterfall model.

- Testing: Testing is the fourth phase of the waterfall model. Once, the system is ready, it is deployed or hosted in a testing environment. The testing of the system is carried out to verify functional and non-functional requirements are met which is set by the customer during the requirement analysis phase. During the testing, the possible bugs and defects are tracked and reported back to the developers, and fixed. The system is tested multiple times and the process continues until the application is stable, bug-free as much as possible, and covers all the business requirements.

- Deployment: In this phase, the application is deployed or packaged for installation to the environment where the end-users can start using it. The system is monitored for any possible deployment issues.
- Maintenance: After the deployment or release of the system, the end-users will start using it. During this phase, three major activities are performed: Bug fixing, upgrade, and enhancement. Feedbacks are taken from the end user's experience and any bugs that are tracked will be fixed. It ensures the system is consistently performing as per the specification.

![Waterfall Model](image)

Figure 2. Waterfall model [6]

4.2 Agile Methodology

Agile methodology is one of the project management methodologies, it breaks down the development into multiple iterative cycles. Each cycle lasts one to three weeks and involves team collaboration across multiple dimensions of project development such as Planning, Requirement Analysis, Design, Implementation, Testing, and Delivery continuously. This approach allows the project to be adjusted and remodeled easily and reach the goal with updated requirements that fits the latest business need.
The hybrid method which is a combination of the Waterfall method and Agile method shown in figure 4 below was used while coding the application. This provided benefits of both methodologies and to iterate the logic, design, and features of the application much conveniently.

4.3 Flow chart of the application logic

The flow chart below in Figure 5 describes the logic of the application. It begins with opening the application, then the user can see customizable settings in the application UI. They can be changed, or the default setting can be used in which case it will generate the steps of sorting for the default sorting algorithm. To visualize the sorting process, all navigation buttons except the “reset” button can be pressed or the “start” button for
automatic visualization. The reset button will generate the random array again. After the visualization is complete, the process can be repeated, or the application can be closed.

Figure 5. Application Flow chart

5 Web application as a Visualization tool

This section lists all the tools that were used in the project to create the visualization tool with a web application. A brief definition of the tools includes their history, usage, and their role.
5.1 HTML, CSS

HTML was introduced in 1993. It was developed by Tim Berners-Lee. HTML is a short form for Hypertext Markup Language. A markup language is a computer language used to create documents that are human-readable using elements that are expressed as tags. HTML is one of the most popular markup languages. It is used to build web documents or web pages and describe their structure. HTML provides structure to the webpage using its elements or tags. HTML can be written in any basic text editor. The content of the HTML file is stored in the file with the extension “.html” or “.htm” which is shown in figure 6 below. The structure of a basic webpage written in HTML is shown in the figure below. The file can be opened with a web browser that uses an HTML interpreter to parse and display the file on the webpage. From the first release, there have been 5 different versions of HTML. The latest one is called HTML5. [8]

![HTML file with basic syntax and its output](image)

CSS stands for Cascade Style Sheets. It is a tool used to add presentation styles such as colors, layout, animations, and fonts to the web pages. It allows the web pages to adapt to the different screen sizes across various devices. CSS is independent of HTML thus it can also be used with any XML-based markup language. CSS uses a selector to target the HTML element and add styling to them. The CSS properties can be placed inside the `<style></style>` tag inside the `<head></head>` tag of HTML file or it can be separated into a different file with “.css” extension. Figure 7 below shows a basic HTML file with CSS for styling.
5.2 JavaScript and TypeScript

JavaScript is one of the most popular scripting languages at the current time. Though, widely popular as JavaScript, its official name is ECMAScript. It is a lightweight, interpreted, or just-in-time (JIT) compiled programming language. It was initially created by Brendan Eich and first released on the year 1995. [9] Currently, it is standardized and maintained by ECMA International. The file extension for JavaScript is “.js”.

TypeScript is a language built on top of JavaScript with additional features that are initially unavailable. In general, TypeScript is a superset of JavaScript. TypeScript provides features such as static type checking, types, interfaces, immutable and read-only objects, etc. TypeScript adds strictly typed language experience to JavaScript. But JavaScript engines cannot run TypeScript so, the code written in TypeScript is compiled into JavaScript by tools like typescript compiler or babel first and then it can run in a browser, Node.JS, or in any JavaScript app. The codes are written in a file with a “.ts” extension, any valid JavaScript code is also TypeScript code, but TypeScript’s type checking will point out errors that might cause possible type mismatch and help minimize run time errors. The main aim of TypeScript for building a JavaScript application is to avoid a potential syntactical error that may cause runtime errors earlier during the development phase. TypeScript also helps bring a proper structure and documentation to the
codebase and scalability to the whole project. TypeScript is developed and maintained by Microsoft. [10]

Figure 8 differentiates between the syntax of JavaScript and TypeScript. It also includes the type checking feature of TypeScript.

![JavaScript and TypeScript syntax comparison](image)

Figure 8. JavaScript code vs TypeScript code

5.3 React

From the Reactjs.org official website, "React is a JavaScript library for building user interfaces". It can be further elaborated as a frontend JavaScript library for building a single-page application (SPA) with reusable and declarative component-based user interfaces. It was created by Facebook Inc. and released for the first time in 2013. Since then, it has become the most popular library to build the frontend at the time of writing the thesis. [11]

React uses HTML like JavaScriptXML (JSX) syntax to create a component thus, making it easier to learn. An example of a simple component is given in figure 9. Every DOM element is written as a React component. The component can be reused to render alike components which reduce the repetition of code. Components can also render single or multiple components inside them and act as a node in a DOM tree. With the traditional web development approach, a website consisted of multiple static HTML files for each
page in the website. The pages were separated based on route or URL path and the website had to reload each time to load a new page or to show a new change in the UI. Unlike traditional web development, React consists of a single HTML page hence, the application is called Single Page Application (SPA). Furthermore, React renders a Virtual DOM which is stored in the memory. In case of any state changes, the changes are compared to the Virtual DOM in the memory and only related elements (along with their children component) are updated in the actual HTML page (or DOM). Everything in the application is dynamically rendered on the same single page thus, eliminating the need to refresh the page to load new content enhancing faster performance and a pleasant user experience. [12]

![Basic React Component](image)

**Figure 9. Basic React component**

5.4 Other tools

This section includes all other tools that were part of the application development. It describes them in brief detail with their usage.

5.4.1 Node.js

Node.js is a cross-platform JavaScript runtime environment, that allows JavaScript to run in the server-side platform. Node.js was developed by Ryan Dahl in 2009 [13]. It is built using Chrome’s V8 JavaScript engine with additional feature-rich JavaScript libraries. It can be used to develop server-side and networking applications. By design, all APIs of Node.js is asynchronous which means, the server doesn’t wait for the task to finish. it can handle new calls to the API without blocking (non-blocking) before returning the
result of the previous API call. It is very fast in code execution especially the time to load the server for the first time (cold start). It is single-threaded, non-buffering and it is open source with contribution from a large community. [13]

5.4.2 Redux

Redux is a state management tool for JavaScript applications. It is used to centralize the application state and provide logic to access and modify the state. It helps the application to behave consistently and has cross-environment support such as client, server, and native. It is based on flux architecture (figure 15) which describes the unidirectional flow of data in an application, the application state is separated from the UI into stores. Any changes to the state are triggered via which is dispatched by dispatchers, the data will update and finally displayed in the UI. The actions could be triggered by user interaction with the UI e.g., typing in a textbox or pressing a button. [14]

![Flux Architecture Diagram](image)

Figure 10. Flux architecture [15]

5.4.3 React Router

React Router is a navigation component library that enables navigation or routing among components in a React application. By design, React applications are single-page applications. With the use of React Router, any React application can behave as a multi-page application. It allows the React applications to change views based on browser URL and path. It is also available for React-native applications. [16]
5.4.4 Sass

Sass is a pre-compiler stylesheet language that is interpreted or compiled into a CSS stylesheet. Sass also provides easy-to-use features such as variables, nested rules, mixin, functions, and more that are compatible with CSS. Sass shares similar syntax to CSS and helps keep styling for larger applications well-managed and facilitates sharing design more easily. The file extension for a Sass is ".scss". [17]

5.4.5 Webpack

Webpack is a module bundler for JavaScript applications. It takes all the JavaScript modules of the application, and their dependencies then bundle them into one or more bundles (files). The bundling is supported for both frontend and backend JavaScript applications. It can be used for code splitting which allows to split the code into various bundles and control the priority of loading them to optimize the load time. Along with JavaScript codes, webpack can be used to bundle and minify other files such as CSS, SVG, image, and other many files type with the help of a loader. [18]

5.4.6 Babel

Babel is a JavaScript compiler tool to convert modern JavaScript especially ECMAScript 2015+ code to older or backward-compatible versions of JavaScript. Modern JavaScript is converted into an older version of JavaScript which is widely supported across several browsers or environments. It plays important role in keeping the use of the latest JavaScript feature in the codebase and still maintains the support for older browsers. Among many other features, babel’s primary feature also consists of transforming the syntaxes, polyfill features that are not yet available in the target environment, transforming the source code, and more. [19] Figure 11 shows the ES2015 version of JavaScript being converted to the ES5 version.
5.4.7 GitHub and Git

GitHub is a web interface for version control based on Git. Git is an open-source version control software. It allows a team of several people to work continuously on a single project and make changes to the same file. GitHub can be used to store the files and collaborate with teams efficiently. Along with version control, GitHub provides other services such as CI-CD, team collaboration tool, etc.

6 Project Implementation

This section will describe the project implementation, coding practices, vision of the project in detail.

6.1 Building the application

The goal of the project was to create a visualization platform to visualize various sorting algorithms in the form of a web application. There are abundant choices of tools for creating web applications. But the tools mentioned in the previous section are the ones used while creating a project.

A scalable project requires a strong project setup; thus, the project began by creating a robust project setup that contains the tooling required to run the project. The setup was
built with tools like babel for converting modern JavaScript to compatible versions for various browsers, webpack for bunding the code into a bundle file, TypeScript for type checking, SASS for precompiled CSS with extra features and React. Git for version control, ESLint for checking the codes for syntactical consistency. A proper folder structure was created, and separate folders were assigned for its purposes. The folder structure is described in the section below. Once, the base is ready, a repository was created in GitHub and the files were pushed into it. The base project can be used to create various other React applications and would reduce the tasks that are often repeated while initializing a new project.

After the template is created, the actual implementation of the project that was collected as requirements during the Requirement analysis phase of the project was started. Here, the Hybrid development method was used. The features were split into numerous small tasks and the Trello application was used to track the progress of the tasks. This gave an updated overview of the progress of the application. While this approach is time-consuming when working in a small team, it was still helpful to organize and keep the progress steady and continuous. And it is helpful to guide the project on a larger scale with more team members across various aspects of the project.

6.2 Implementing the visualization in React

React is one of the most popular JavaScript frontend frameworks/libraries on the Stackoverflow survey at the time of writing this thesis. [13] It is based on the idea of reusability of components which allows faster development of application with it. For the same reason, it is used as the frontend library for this thesis project.

React provides full control over the component and its lifecycle method (also called hooks). There are three different phases of lifecycle namely Mounting, Updating and, Unmounting. Mounting and unmounting happen while the component is being loaded and removed from the memory respectively. Updating happens when the state changes in the component or of its parent component. Lifecycle hooks provide a way to call customized functions during these phases.
As there are two different types of React components: Class component and Functional component. Each of them has different APIs to access the lifecycle methods shown below in figure 12. This thesis project uses Functional components and its lifecycle hooks of React.

---

### React Lifecycle Methods/Hooks

Comparison of React lifecycle methods in Class Component and their equivalent hooks in Functional Component.

<table>
<thead>
<tr>
<th></th>
<th>Class Component</th>
<th>Functional Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mount</td>
<td><code>componentDidMount()</code> { }</td>
<td><code>useEffect(() =&gt; { // runs on mount (only once) }, [])</code></td>
</tr>
<tr>
<td>Update</td>
<td><code>componentDidUpdate()</code> { }</td>
<td><code>useEffect(() =&gt; { // runs on every update })</code></td>
</tr>
<tr>
<td></td>
<td><code>shouldComponentUpdate() { }</code></td>
<td><code>useEffect(() =&gt; { // runs if any dependency changes }, [dependencies])</code></td>
</tr>
<tr>
<td>Unmount</td>
<td><code>componentWillUnmount()</code> { }</td>
<td><code>useEffect(() =&gt; { return () =&gt; { // runs on unmount (only once) } })</code></td>
</tr>
</tbody>
</table>

---

Figure 12. React lifecycle methods

The application generates an array of unique and random numbers of different sizes/lengths which is also adjustable using the menu in the UI. Then the array of numbers is mapped into a TypeScript type of “Bar” with keys or properties value and status. The value key has the value of one of the numbers from the array and the initial value for the key status is UNSORTED. The possible values for the status property are UNSORTED, SORTED, ACTIVE, PIVOT. While the status key is used to give different colors to the Bar object, the value is used as the height of a vertical column that is displayed
in the UI which is elements shown in figure 13. The color of the Bar visualizes the status of the element in the array.

Figure 13. Visualization of Quicksort algorithm and legends

The mapped array of Bar from the random array of numbers is then passed to the selected algorithm to sort them. Based on the design of the sorting algorithm, it iterates over the array, compares, and swaps the position of the elements or Bar. When two elements are taken for comparison, their status is changed to ACTIVE, when the element reaches its final position then it is changed to SORTED otherwise it will be UNSORTED. In the Quicksort algorithm, a pivot is taken in the middle of the array for comparison of elements on its two sides hence, the element in the index of pivot will have the status of PIVOT. The status and position of elements change multiple times over different iterations.

The application uses an array called steps to keep track of all the changes that take place in the original random array in different phases of the sorting. With every change, a new copy of the changed original random array is pushed into the steps array. A change can be the change of status of elements or swap of position among two elements. The steps are then used to visualize the sorting of the randomly generated array.

As mentioned in the paragraph above React updates the UI when the state or props of the component change. The application uses a component called BarContainer that is
responsible for displaying the array in the UI. The component maps the array into another component called BarComponent. The BarComponent then converts the value and status of the Bar or element into a vertical column in the UI. The BarContainer takes a step (which is the initial or modified array of the random number) as its prop. The application utilizes setTimeout with an interval to pass a different step as a prop to the BarContainer which will make it re-render the vertical columns in the UI and display the changes the sorting algorithm made. The interval is used to increase or decrease the speed with which a new step is passed as props to the BarContainer component. This will simulate a visualization of the steps an algorithm takes to sort a given array.

While this approach works, the implementation is still not optimal. With every new step passed as a prop to the BarContainer component, all its children i.e., all the BarComponents are re-rendered even if only a few elements of the array were changed between the steps. In a larger application, this can bring performance issues. React provides a method called memoization to overcome this issue. We can wrap the component inside React.memo() hook and it will memoize the component. Once, a component is memoized, it will prevent it from re-rendering even if its parent component is re-rendering while its props have not changed. This approach helps boost the application performance.

Figure 14. No. of re-rendering before and after memoization using Bubble sort
6.3 State Management

State management is one of the important parts of a React application. Better state management in a React application leads to refined and easy-to-scale applications. React provides, useState hook for internal state management for a functional component and context for global state management. While these solutions work on a smaller scale, as the application grows the complexity of the state management grows with it. Thus, this thesis project uses Redux a popular JavaScript state management library that simplifies the application by separating the logic of the state and the UI. Redux provides hooks to retrieve and update the state. The components that require the state or the update method use those hooks.

The application uses Redux to keep track of various states that are used throughout different components. The states used in the application are listed below:

- animation or visualization speed
- array size
- list of algorithms
- selected algorithm
- initial random array
- steps of sorting the array
- current step to be displayed in the UI

Along with these states, Redux provides methods called actions to update the state. The actions are used to update the array size, animation speed, select a new algorithm, navigate to previous and next steps, and reset the sorting.

In this thesis project, Redux helped to isolate the application’s state logic from React’s UI logic. Implementation of Redux was done during the latter phase of the project. While it added an extra module to the application, it brought a proper structure to the whole application logic and introduced harmony between the components and the state management. The application also became more scalable and easier to modify.
6.4 Project structure

For any software development project, the structure and arrangement of files and folders provide a foundation for scalability and maintainability. And it is a key yet sometimes overlooked aspect that should be priorities during software development phases. The layout of the project structure helps anyone new to the project to get familiar easier and quicker. As the application grows, it keeps the project files in logical order so, the components, utility methods, styles, etc. will grow consistently and still be manageable. The approach for a modular and scalable project structure will also assist greatly in finding bugs and debugging the application. Numerous approaches are widely practiced in various application development standards. Yet, there is no single optimal solution. It also depends on the type of project and its requirement.

Here is a table with the screenshot (Figure 15) along with the role of the folders of the project structure in this thesis project. The files are arranged in folders with names that describe the purpose and its contents.
6.5 Hosting

The final part of web application development is hosting. There are numerous popular providers like Amazon Web Services, Google Cloud Platform, Azure, etc. that provides the solution for hosting web application. These platforms provide a multitude of features on top of hosting at an affordable price. For this thesis project, a web hosting service called Netlify was used. It is a very straightforward, fast, and easy-to-use web hosting service that also supports free web hosting. It provides a cli that can be used to deploy
the application from the terminal. In addition to free web hosting, it also provides a custom domain name with the suffix of “netlify.app”. The application is hosted with netlify in the address https://algo-react-viz.netlify.app.

7 Project outcome

By the end of the thesis, and with numerous iteration a web application visualizing three sorting algorithms was successfully created using React. In the UI, the options are available to choose the preferred size of the input array, sorting algorithm, visualization speed, and color mode. Along with the options, a list of control is also added to the UI to navigate and reset the steps of sorting. The SPA nature of the application allows the changes to be visible without reloading the page. The final UI of the application is as in Figure 16 below.

Figure 16. UI of the application

Figure 17 below shows visual differences between various sorting algorithms with the same set of data. The difference in visualization shows the approach each algorithm takes to sort the data. The merge sort and quick sort algorithms are based on the divide and conquer strategy while the bubble sort is based on the iterate and compare strategy.
While the visualization using colors to denote the status of the elements is more noticeable, it arises concern of accessibility for users with medical conditions that may be affected by rapid changing or flashing colors. [20] Thus, to accommodate such a case, a cautionary warning is prompted when the application is opened for the first time and the color mode feature of the application is turned off by default. Figure 18 shows the difference in visualization with color mode turned on and off.

Figure 17. Visualization of the sorting process in different sorting algorithms

Figure 18. Merge sort visualization in different color modes
Along with main sorting visualizer UI, the application consists of two more pages i.e., Algorithms and About. The algorithms page (Figure 20) contains the view with table of content which is description of all the topics that are part of the theoretical background of the thesis report. On the left of the UI, it consists of the menu which is made up of links created with a tag and will scroll the page to the respective section on clicking them. Each section on the right side of the UI contains a header with an “id” attribute which is passed as “href” attribute on the side menu.

**Algorithm**

Algorithm can be defined as a set of steps to accomplish a given task. Its purpose is to make it easier to replicate doing the task again and be efficient and accurate to the same degree. The term 'Algorithm' is derived from the name of 9th century Persian scientist, astronomer, and mathematician Abu al-Àla al-Khwarizmi who is also regarded as 'The Father of Algebra'.

Algorithm can be found everywhere in day-to-day life. The primitive form of algorithm began with calculating and at the present time, it is being used in complex study and operation such as artificial intelligence, computer processing, gaming, molecular biology, and various other scientific research areas. It can be as simple as steps to turn on and off a light switch to any mathematical or computational operations. It is used primarily for calculation, data processing, and automated reasoning. The importance of algorithm arises once quality and efficiency are key factors to be considered while accomplishing any tasks. Opposite of following an algorithmic process is performing the same task, each time in different and unique manner. This introduces unreliability, inefficiency and increases time, energy, cost, and many other factors. Thus, use of algorithm helps eliminate these hurdles, brings consistency, efficiency and quality which is visible as the result.

Algorithm is related to function and sometimes they are even used interchangeably but they are still not same. While an algorithm is a set of ideas or an abstract concept that elaborates on how to solve a problem, and functions are the actual implementation of the algorithm which can be used to solve them. Hence, a function can be an implementation of a whole algorithm or a single or multiple steps of an algorithm.

**Sorting Algorithms**

![Figure 20. Algorithm page](image-url)
The last page is the About page which is shown in figure 21 below. It contains a brief description explaining the intent of the thesis project, technologies used, and the link to the GitHub repository.

![Figure 21. About page](https://github.com/bkrmadya/sorting-algorithm-visualizer)
8 Conclusion

The goal and main objective of the thesis were to bring sorting algorithms and visualization together. The concept of sorting algorithm mostly expressed through codes and syntax was still abstract. Thus, this thesis was an attempt to eliminate that by creating a tool that can visualize the process and steps of sorting algorithms. Visualization is a form of expression; it can take numerous forms to convey the idea, yet it always provides a new outlook if it is implemented properly. This thesis report can be taken as a visualization of the objective and idea of the thesis in the form of report writing with letters, figures, and tables. This thesis project was successful in delivering the idea of visualization of sorting algorithm through a web application and providing a separate look into the algorithm apart from lines of code and syntaxes.

The tools, language, and approach taken in this thesis project are not crucial or imperative, but they are just means utilized to visualize or execute the idea behind the thesis and its objective. They are popular tools and choices for web application development at the time of writing this. There is plenty of space for improvements which can be achieved through better UI design, accessibility, application logic, performance, and/or tools.

Algorithms are an integral part of programming and computer science. Visualization is the means to communicate. Those abstract ideas of algorithms can attract more people to learn and understand them if they are visualized in more interactive ways.
References

1. History of Algorithm URL:  

2. Analysis of different sorting techniques URL:  
   https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/  
   [Accessed July 2021]

3. What is Big O notation explained URL:  

4. Top 10 oldest art ever discovered URL:  

5. World’s oldest known cave painting URL:  

6. SDLC - Waterfall model URL:  

7. Agile and Waterfall – Benefits of Blending the Two URL:  

8. HTML History URL:  https://www.w3schools.in/htmltutorial/history/ [Accessed July 2021]

9. JavaScript History URL:  https://www.w3schools.com/js/js_history.asp  
   [Accessed July 2021]


