# Acknowledgement

The warm response received by our proposal, both from teachers and supervisor, has encouraged the learning group to present our ideas and vision within time. Every topic of this proposal has been written in simple and lucid language so that anyone going through it will be clearly able to understand the subject. Similarly, the context has been revised thoroughly to make it more comprehensive.

We would like to express our sincere gratitude to our supervisor **Additional Associate Professor Udhhav Bhhatarai** for his guidance and support throughout the work. Our deepest appreciation to the entire team of Cosmos College of Management and Technology, for granting us this valuable opportunity and helping us reach our milestone by giving valuable feedback and suggestions. For the improvement of the subject matter of this proposed project, constructive criticism, suggestions, and feedback for the improvement of this project would be highly appreciated.

**- Team AVT**
Gajananda Adhikari
Nischal Khanal

# Abstract

Development of flow-charting and animations clearly suggest in improved comprehension and teaching of various algorithms. Current research in software visualization has resulted in the development of algorithm animation systems allowing students visualize the workflow of the algorithm. This article studies the effectiveness of such algorithm visualization tools under various conditions and settings. In response, this project proposes an extended approach to AV, termed Algorithm Visualization Tool (AVT), aimed at enhancing algorithm comprehension through improved retention and recall and collaborative and independent learning. By integrating dynamic parameter selection into the AVT, students can gain insights into the selection of algorithms based on specific problem scenarios, thereby advancing their understanding to higher cognitive levels. This project aims to contributes to bridging the gap between algorithm visualization and comprehensive algorithm understanding, paving the way for more effective algorithm education strategies.

Keywords: Algorithm Visualization, Case-Based Performance Comparison, Algorithm Comprehension, Evaluation-Integrated Development, Computer Science Education

# Contents

# List of Figures

# Acronyms

| | |
|---|---|
| **AV** | Algorithm Visualization |
| **AVT** | Algorithm Visualization Tool |
| **UI** | User Interface |
| **CS** | Computer Science |
| **IT** | Information Technology |
| **JS** | JavaScript |
| **API** | Application Programming Interface |
| **DOM** | Document Object Model |
| **IDE** | Integrated Development Environment |
| **UX** | User Experience |
| **MVC** | Model-View-Controller |
| **JSON** | JavaScript Object Notation |

# Chapter 1
# Introduction

## 1.1 Introduction to the Project

In the realm of computer science education, understanding algorithms stands as a fundamental pillar. However, grasping the intricacies of various algorithms often poses a significant challenge for undergraduate students, hindered by the complexity of algorithms and the constraints of traditional classroom settings. To address this issue, our project aims to develop an Algorithm Visualization Tool (AVT), a web-based interface designed to facilitate the comprehension of multiple searching and sorting algorithms. By providing a platform where users can dynamically visualize algorithmic processes and manipulate variables, the AVT seeks to enhance learning experiences and foster deeper understanding among students.

## 1.2 Team Members

The development of the Algorithm Visualization Tool (AVT) project is undertaken by undergraduates i.e. 6th semester BEIT students: Gajananda M. Adhikari - 200111, Nischal Khanal - 200120 to better enhance our understanding the working principle of several algorithms, the visualization tools and techniques for those algorithms and to enhance our skill in Web Development Stack. Our dedication for the project lies in the value of it's educational purpose for the computer science students as well as ours. We hope to have understood the workings of at least searching and sorting algorithms with the completion of the project.

## 1.3 Objectives

1. Develop an intuitive web interface, the Algorithm Visualization Tool (AVT), for real-time visualization and manipulation of searching and sorting algorithms.

2. Integrate case-based performance comparison into the AVT to enhance algorithm understanding and usability for undergraduate computer science students.

# Chapter 2
# Literature Review

Algorithm Visualization tools have become crucial aids in teaching algorithms to undergraduate Computer Science students.These tools provide visual representations of algorithms, allowing students to better understand the inner workings and logic behind them [2]. However, existing AVs often fall short in providing a comprehensive understanding of algorithms beyond the application level due to their focus solely on visualizing algorithm steps [8]. To address this limitation, we propose an extended approach to AV, termed Algorithm Visualization Tool [2], aimed at enhancing algorithm comprehension through case-based performance comparison [4].By integrating case-based performance comparison into the AVT, students can gain insights into the selection of algorithms based on specific problem scenarios, thereby advancing their understanding to higher cognitive levels [3]. Additionally, the AVT adopts an evaluation-integrated development approach, allowing students to not only visualize the algorithms but also evaluate their performance and compare them withother algorithms.This approach will provide a more comprehensive learning experience, allowing students to not only understand the algorithms conceptually but also apply critical thinking and analytical skills in selecting the most appropriate algorithm for a given problem. Furthermore, the AVT will provide a platform for students to actively engage in the learning process by allowing them to construct their own visualizations.

Our proposed AVT builds upon these approaches by integrating case-based performance comparison and evaluation-integrated development. By providing students with a platform to visualize, evaluate, and compare different algorithms, the AVT aims to provide a more comprehensive learning experience. This approach will enable students to not only understand algorithms conceptually but also apply critical thinking and analytical skills in selecting the most appropriate algorithm for a given problem.

Furthermore, the AVT will provide a platform for students to actively engage in the learning process by allowing them to construct their own visualizations. This hands-on approach will enable students to deepen their understanding of algorithms and develop problem-solving skills. By comparing the performance of different algorithms in various scenarios, students can gain a better understanding of the strengths and weaknesses of each algorithm.

In [7], the paper ”A User-Centred Approach for Designing Algorithm Visualizations” presents a user-centered design approach for algorithm visualizations. This approach emphasizes usability testing and iterative design, ensuring that the visualizations meet the needs of different users. The visualizations are designed to be simple, flexible, and customizable, while the tool includes interactive features such as zooming, panning, and filtering. Users are also able to create their own visualizations using the provided templates, making the tool a valuable resource for understanding and reasoning about algorithms, with an overall aim to support the learning and teaching of algorithms.

The paper ”Design and Evaluation of a Web-based Dynamic Algorithm Visualization Environment for Novices” [9] describes a dynamic algorithm visualization tool for novice users. It features a user-friendly interface with interactive visualizations that allow users to manipulate input data and observe algorithm behavior in real-time. The tool also includes essential interactive features such as step-by-step execution, pause and resume, and reset. Users can customize the visualizations by adjusting colors and shapes. This environment is designed to facilitate self-paced learning, making it easier for novices to comprehend and learn algorithms.

GAVEL, introduced in the paper ”GAVEL - a New Tool for Genetic Algorithm Visualization” [1], offers a graphical user interface and interactive visualizations for genetic algorithms. The visualizations in GAVEL are clear, concise, and informative, providing users with an overview of the genetic algorithm’s progress and performance. Users can select and compare different visualization modes and customize the parameters, which enhances their ability to understand and analyze genetic algorithms. The tool is aimed at supporting both research and development in the field of genetic algorithms.

”Visualization Tools for Real-time Search Algorithms” [5] presents tools specifically designed for real-time search algorithms. The visualizations are dynamic and responsive, providing real-time feedback on the behavior and performance of the search algorithms. Users are able to interact with the visualizations by zooming, panning, and filtering, as well as customizing the display using different colors and shapes. These features make the tool highly useful for understanding and optimizing real-time search algorithms, supporting research and development in this area.

Finally, the paper ”Algorithm Visualization in CS Education: Comparing Levels of Student Engagement” [6] evaluates the effectiveness of visualization tools in computer science education by comparing different levels of student engagement. The visualizations are designed to be engaging, interactive, and informative, offering an intuitive understanding of algorithms. With features that allow students to explore, manipulate, and compare different algorithms and visualization modes, the tool aims to enhance learning and engagement in CS education.

# Chapter 3
# System Analysis

## 3.1 Current Challenges

In developing the Algorithm Visualization Tool (AVT), it's essential to recognize the current challenges faced in computer science education regarding algorithm comprehension. Traditional methods of teaching algorithms often rely heavily on theoretical explanations and limited practical demonstrations, leading to difficulties for undergraduate students in grasping complex algorithmic concepts. Furthermore, the lack of interactive learning tools tailored specifically to algorithm visualization poses a significant obstacle to effective learning. Without dynamic visualization capabilities, students may struggle to understand the inner workings and practical applications of various algorithms.

Additionally, the absence of comprehensive performance comparison tools impedes students' ability to discern the most suitable algorithm for specific problem scenarios. Understanding the comparative advantages and limitations of different algorithms is crucial for effective problem-solving and algorithm design. Without access to such tools, students may find it challenging to apply theoretical knowledge to real-world problems, limiting their overall proficiency in algorithmic thinking and problem-solving skills.

## 3.2 Requirement Analysis

To address the current challenges in algorithm education and enhance student learning experiences, the Algorithm Visualization Tool (AVT) must meet specific requirements to ensure its effectiveness and usability:

### 3.2.1 Interactive Visualization

The AVT should provide an intuitive and interactive visualization interface that allows users to observe the step-by-step execution of various searching and sorting algorithms. The visualization should be dynamic, allowing users to manipulate input variables and observe how changes affect algorithm behavior in real-time. By providing a visual representation of algorithmic processes, the AVT aims to enhance students' understanding of algorithmic concepts and promote active engagement in learning.

### 3.2.2 Case-Based Performance

Incorporating case-based performance features into the AVT is essential for enabling students to compare and contrast the performance of different algorithms under various problem scenarios. The AVT should allow users to input different datasets and compare the execution times, space complexities, and other performance metrics of multiple algorithms. By providing comparative insights, the AVT empowers students to make informed decisions when selecting algorithms for specific problem-solving tasks, thereby promoting higher-order thinking skills and algorithmic fluency.

### 3.2.3 User-Friendly Interface

The AVT should feature a user-friendly interface that is accessible to undergraduate computer science students with varying levels of programming experience. The interface should be intuitive and easy to navigate, allowing users to quickly access and interact with different algorithms and visualization features. Additionally, the AVT should provide clear instructions and guidance to help users understand how to use the tool effectively and maximize their learning outcomes.

### 3.2.4 Evaluation and Feedback Mechanisms

Implementing evaluation and feedback mechanisms within the AVT is crucial for assessing its effectiveness and gathering user input for continuous improvement. The AVT should include built-in evaluation tools that allow instructors to track student progress, monitor usage patterns, and identify areas for improvement in algorithm education. Additionally, the AVT should solicit feedback from users through surveys, user testing sessions, and other feedback channels to ensure that it meets the evolving needs of students and educators.

# Chapter 4
# System Development

## 4.1 Interface Development

The development of the Algorithm Visualization Tool (AVT) follows the Incremental Waterfall Model, combined with Scrum practices to allow for iterative development and frequent feedback. This hybrid approach ensures that each stage of development is thoroughly planned and executed while remaining flexible to incorporate changes and improvements.

### 4.1.1 Phase 1: Requirements Analysis

During the initial phase, we plan to conduct a comprehensive analysis of user requirements and system specifications. This involves gathering input from stakeholders, defining functional and non-functional requirements, and establishing the scope of the project. Based on this analysis, the team creates a detailed requirements document outlining the objectives, features, and constraints of the AVT.

### 4.1.2 Phase 2: Design

In the design phase, we plan to focuse on creating the user interface (UI) and interaction design for the AVT. This involves wireframing, prototyping, and creating mockups to visualize the layout and functionality of the application. The design process incorporates feedback from stakeholders and usability testing to ensure that the UI meets the needs of its intended users.

### 4.1.3 Phase 3: Implementation

Once the design is finalized, we plan to proceed with the implementation phase, where they translate the design specifications into working code. The AVT is developed using modern web development technologies, including Node.js for server-side scripting and React.js for building dynamic user interfaces. JavaScript, CSS, and HTML are used to add interactivity and styling to the application.

### 4.1.4 Phase 4: Iterative Development

The development process is structured into iterations, each focusing on visualizing a specific algorithm. This Scrum-like approach ensures continuous delivery and allows for regular feedback and improvements. The steps for each iteration are as follows:

- Iteration Planning: Define the scope and objectives for the current iteration, focusing on a single algorithm.
- Development: Implement the visualization for the selected algorithm within the web interface.
- Testing: Conduct rigorous testing to identify and resolve any issues with the new visualization.
- Review: Gather feedback from stakeholders and users to assess the effectiveness of the visualization.
- Deployment: Deploy the updated version of the AVT with the new algorithm visualization.

This iterative process is repeated for each algorithm, gradually expanding the functionality and scope of the AVT.

### 4.1.5 Phase 5: Feature Enhancement

After the initial visualization of algorithms, additional features such as parameter control and performance comparison between algorithms will be developed. These features will also follow the iterative development process to ensure they are well-integrated and functional.

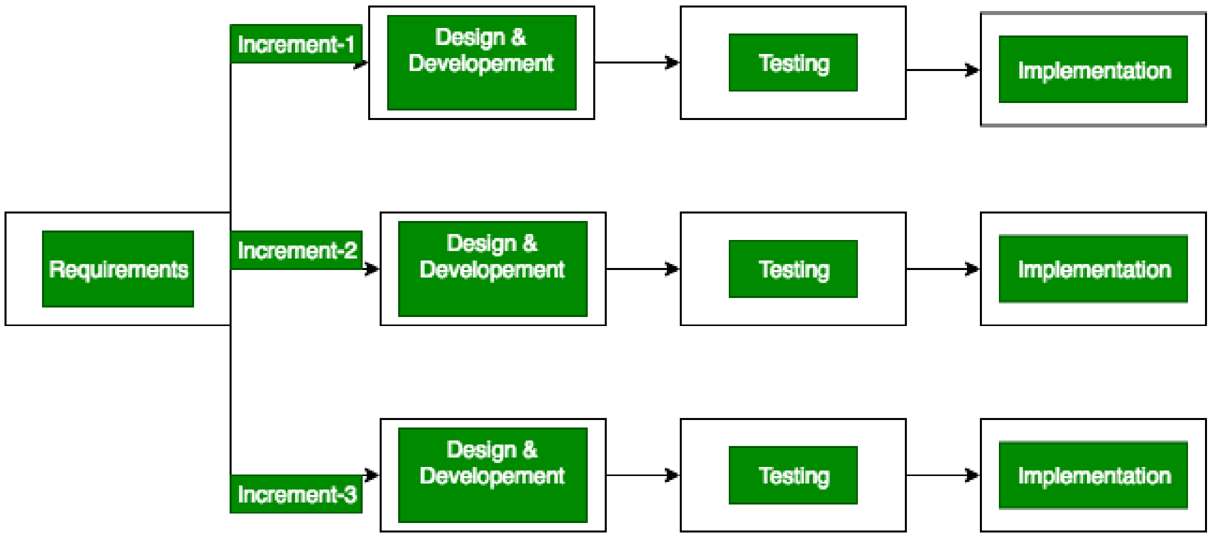


Figure 4.1: Incremental Waterfall Model

## 4.2 Technology Stack

The Algorithm Visualization Tool (AVT) is built using a minimal tech-stack. We have focused on scheduling algorithms and used Bootstrap elements for the visualization tool. This allows us to create tables and graphs for better visualization. We have completely opted for HTML, CSS, and JavaScript, and used Bootstrap for a clean UI along with Canvas functionality.

- HTML: The standard markup language for creating web pages.
- CSS: A stylesheet language used for describing the presentation of a document written in HTML.
- JavaScript: A programming language used to create dynamic and interactive effects within web browsers.
- Bootstrap: A front-end framework for developing responsive and mobile-first websites.

The technology stack is chosen for its flexibility, scalability, and performance, enabling the development of a robust and user-friendly algorithm visualization tool.

## 4.3 Project Timeline

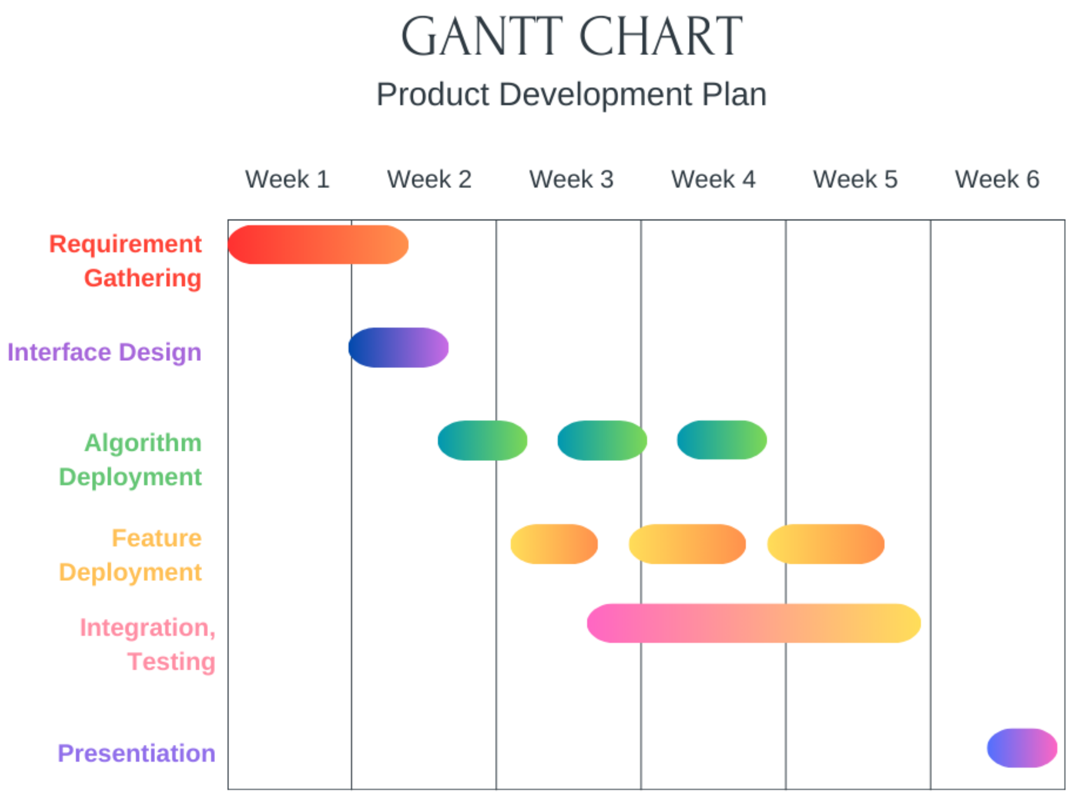
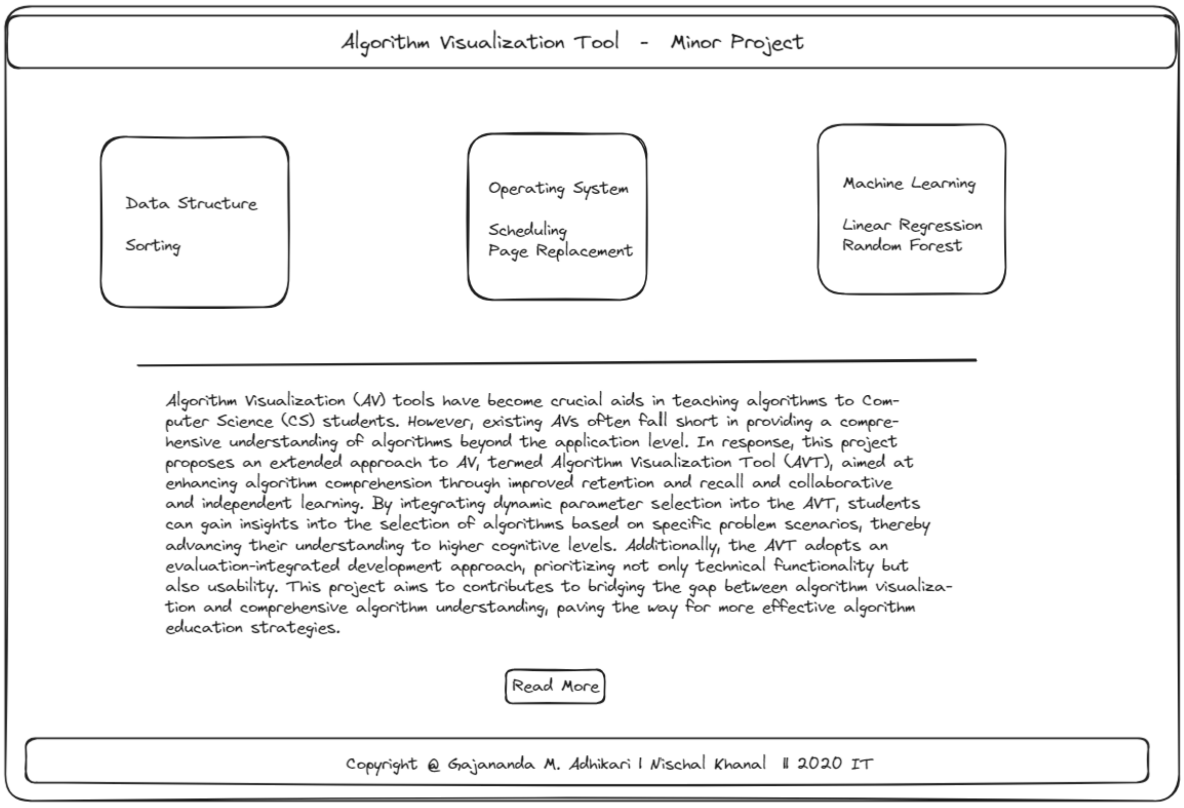


Figure 4.2: Estimated Timeline

# Chapter 5
# Project Description

## 5.1 Interface Design

We have designed the web interface for our project using Excelidraw and Figma. Here are our designs.



Algorithm Visualization (AV) tools have become crucial aids in teaching algorithms to Computer Science (CS) students. However, existing AVs often fall short in providing a comprehensive understanding of algorithms beyond the application level. In response, this project proposes an extended approach to AV, termed Algorithm Visualization Tool (AVT), aimed at enhancing algorithm comprehension through improved retention and recall and collaborative and independent learning. By integrating dynamic parameter selection into the AVT, students can gain insights into the selection of algorithms based on specific problem scenarios, thereby advancing their understanding to higher cognitive levels. Additionally, the AVT adopts an evaluation-integrated development approach, prioritizing not only technical functionality but also usability. This project aims to contributes to bridging the gap between algorithm visualization and comprehensive algorithm understanding, paving the way for more effective algorithm education strategies.

Figure 5.1: Home Page

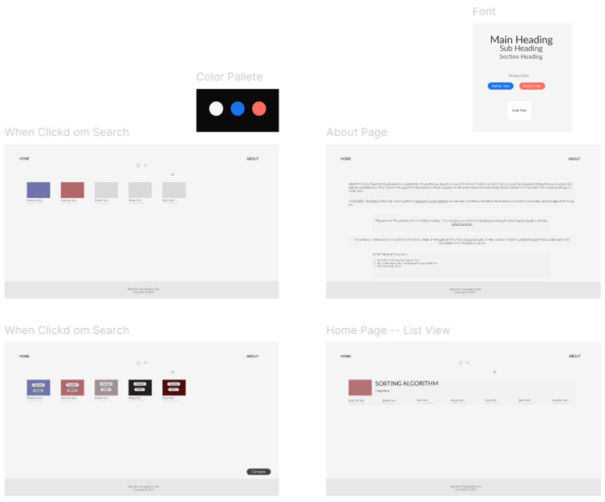
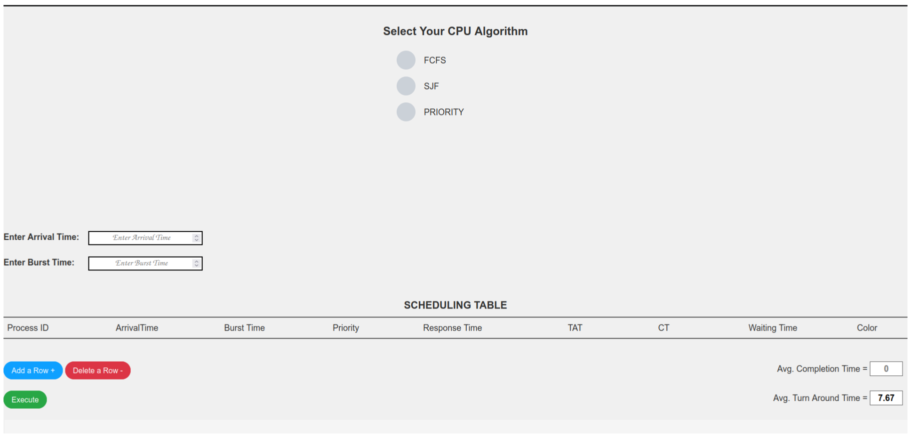


Figure 5.2: Sorting Page design


Select Your CPU Algorithm

FCFS

SJF

PRIORITY

Enter Arrival Time: *Enter Arrival Time*

Enter Burst Time: *Enter Burst Time*

SCHEDULING TABLE

| Process ID | ArrivalTime | Burst Time | Priority | Response Time | TAT | CT | Waiting Time | Color |
|---|---|---|---|---|---|---|---|---|

Add a Row + | Delete a Row -

Execute

Avg. Completion Time = 0

Avg. Turn Around Time = 7.67

Figure 5.3: Scheduling Page design

## 5.2 Algorithms

In our project, we have expanded the focus to include three major categories: **Data Structures**, **Operating Systems**, and **Machine Learning**.

### Data Structures

In this section, we focus on **Sorting Algorithms**. These algorithms are essential for organizing and manipulating data efficiently. We have implemented and visualized the following sorting algorithms:

- **Bubble Sort**: Repeatedly swaps adjacent elements if they are in the wrong order.
- **Quick Sort**: A divide-and-conquer algorithm that partitions the data and recursively sorts the sub-arrays.

- **Merge Sort**: A stable sorting algorithm that divides the array in half, sorts each half, and merges the sorted halves.
- **Selection Sort**: Repeatedly finds the minimum element and swaps it with the first unsorted element.
- **Insertion Sort**: Builds the sorted array one item at a time by repeatedly inserting elements in the correct position.

## Operating Systems

In the Operating Systems category, we explore two key areas:

### Page Replacement Algorithms

- **First-In-First-Out (FIFO)**: Replaces the oldest page in memory.
- **Least Recently Used (LRU)**: Replaces the page that has not been used for the longest time.
- **Optimal Page Replacement**: Replaces the page that won't be used for the longest period in the future.

### Scheduling Algorithms

- **First Come First Serve (FCFS)**: Executes tasks in the order they arrive.
- **Shortest Job First (SJF)**: Prioritizes tasks with the shortest execution time.
- **Priority Scheduling**: Executes tasks based on assigned priority levels.
- **Round Robin (RR)**: Tasks are executed in time slices, cycling through all tasks.
- **Shortest Remaining Time First (SRTF)**: Preempts the current task if a new task arrives with a shorter remaining execution time.

## Machine Learning

Machine learning is a critical and fast-growing field in today's world. With artificial intelligence (AI) being a hot topic, people are eager to understand its core concepts. We believe that clear visualization tools can greatly enhance understanding, making abstract concepts more accessible.

Hence, we decided to implement **Linear Regression**

**Linear Regression**: A statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data.
By incorporating Machine Learning into our project, we aim to provide a foundation for students and learners to grasp these critical concepts through hands-on visualizations, making the learning experience more interactive and engaging.

## 5.3 Challenges

To the extent of our project development, we have had few challenges which we have been able to resolve.
The first one being the design aspect. Our intention for the project was to be educational hence distraction free and efficient. Hence we have used minimalist design philosophy with emphasis on neutral colors.
After that, our focus was for the CS student ranging from entry level to advanced level students. Therefore we decided to start with sorting algorithm since they are usually taught at the start of course and widely used in various fields.
For different visualization tools, we had developed them with clear vision however the theme and integration was done as a whole, not utilizing the common theme or libraries.

In coding perspective, we are developing each models in JavaScript without any framework. We have developed these sorting algorithm which work individually. Hence we had to develop different aspects of our project and test them in unit then later integrate them.

## 5.4 Recommendations

Few recommendations for future to make AVT more efficient and effective are:

- Comparison: Ability to compare different aspects of two or more algorithms for same data
- Complexity: Calculating time and space complexity for different algorithm for different data
- UI: Implementing common color coding for home page to the minute details and different feature reperesentation, which would need to take wide color palette in consideration.
- Animation: Integrating d3.js, p5.js and other animations which can be dynamic and controlled by user input parameters.

# Chapter 6
# Conclusion

The Algorithm Visualization Tool (AVT) project is poised to revolutionize the way algorithms are taught and understood in computer science education. By providing a dynamic, interactive platform for visualizing the intricacies of searching and sorting algorithms, the AVT addresses the limitations of traditional teaching methods and enhances student comprehension. Through iterative development and a user-centered design approach, the AVT ensures that students can engage with complex algorithmic concepts in an intuitive and accessible manner.

Our dedicated team of undergraduate IT students, leveraging modern web development technologies, is committed to creating a robust and user-friendly tool that meets the needs of both learners and educators. The incorporation of case-based performance comparisons and iterative feature enhancements will further solidify the AVT as an indispensable resource in the academic toolkit.

Ultimately, the AVT aims to bridge the gap between theoretical knowledge and practical application, empowering students to develop a deeper understanding of algorithms and their real-world implications. We are confident that the successful implementation of this project will make a significant contribution to computer science education and inspire future innovations in the field.

# Bibliography

[1] gavelCollins, JJ. 1999. GAVEL - a new tool for genetic algorithm visualization Gavel - a new tool for genetic algorithm visualization. Proceedings of the 1999 Congress on Evolutionary Computation Proceedings of the 1999 congress on evolutionary computation ( 2, 1331-1338). https://ieeexplore.ieee.org/abstract/document/942528

[2] denny2024computingDenny, P., Prather, J., Becker, BA., Finnie-Ansley, J., Hellas, A., Leinonen, J.Sarsa, S. 2024. Computing education in the era of generative AI Computing education in the era of generative ai. Communications of the ACM67256–67.

[3] farghally2017towardsFarghally, MF., Koh, KH., Ernst, JV. Shaffer, CA. 2017. Towards a concept inventory for algorithm analysis topics Towards a concept inventory for algorithm analysis topics. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education Proceedings of the 2017 acm sigcse technical symposium on computer science education ( 207–212).

[4] fu2002optimizationFu, MC. 2002. Optimization for simulation: Theory vs. practice Optimization for simulation: Theory vs. practice. INFORMS Journal on Computing143192–215.

[5] realtimesearchHernandez, C., Munoz, J. Perez, J. 2007. Visualization Tools for Real-time Search Algorithms Visualization tools for real-time search algorithms. Journal of Intelligent Information Systems292137-155.

[6] shaffer2004iShaffer, CA., Cooper, ML. Alvarado, C. 2004. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement Algorithm visualization in cs education: Comparing levels of student engagement. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education Proceedings of the 35th sigcse technical symposium on computer science education ( 134-138).

[7] usercentredStasko, J., Domingue, J., Brown, MH. Price, BA. 2002. A User-Centred Approach for Designing Algorithm Visualizations A user-centred approach for designing algorithm visualizations. Journal of Visual Languages and Computing132131-154.

[8] upson1989applicationUpson, C., Faulhaber, TA., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J.Van Dam, A. 1989. The application visualization system: A computational environment for scientific visualization The application visualization system: A computational environment for scientific visualization. IEEE Computer Graphics and Applications9430–42.

[9] urquiza2014designUrquiza-Fuentes, J. Velazquez-Iturbide, JA. 2014. Design and evaluation of a web-based dynamic algorithm visualization environment for novices Design and evaluation of a web-based dynamic algorithm visualization environment for novices. Procedia Computer Science27126–135.