A Report on

Disk Scheduling Algorithm Visualization

OS(IT253) Course Project

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

Karella Udayram	231IT032
G Durga Sai Pavan	231IT022
Kalal Pavan Teja	231IT030
Himanshu Mehar	231IT027

IV Sem B.Tech (IT)

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

April 2025

DECLARATION BY THE STUDENT

I hereby declare that the OS Project Report (IT253) entitled **Disk Scheduling Algorithm Visualization** was carried out by me during the even semester of the academic year 2024 – 2025 and submitted to the department of IT, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in the department of Information Technology, is a bonafide report of the work carried out by me. The material contained in this seminar report has not been submitted to any University or Institution for the award of any degree.

Place: ______________________

Date: ______________________

Kalal Pavan Teja
____
(Name of the Student)

<u>CERTIFICATE</u>

This is to certify that the report entitled "**Disk Scheduling Algorithm Visualization**" has been presented by ***Group 5*** students of **IV semester B.Tech (IT)** Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the even semester of the academic year **2024 – 2025.** It is submitted to the Department in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

Place: ________________________________

Date: (Signature of the Coordinator)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In modern computer systems, the hard disk is one of the most used input/output devices. When multiple read or write requests are sent to disk at the same time, the operating system must decide the order in which that happens. This process is known as disk scheduling. It plays a key role in reducing seek time and improving the performance of the system. Several algorithms are available for disk scheduling, such as First-Come-First-Serve (FCFS), Shortest Seek Time First (SSTF), SCAN, C-SCAN, LOOK, and C-LOOK. Each algorithm follows a different approach and produces different results based on the input.

While from the theoretical explanations and textbook examples, students often find it difficult to visualize how disk scheduling algorithms actually work. Understanding head movements, seek time calculation, and performance comparisons can become confusing without practical examples. Most existing learning tools for disk scheduling are either too theoretical or lack interactivity. Some might have complicated interfaces or require installation of some software. In many cases, users cannot compare different algorithms easily on the same input or visualize the performance difference using graphs. Also, features like real-time voice explanation or saving the results for future comparison are usually missing.

The aim of this project is to make learning these disk scheduling algorithms easy and interactive. The system simulates how the disk head moves while servicing requests using animations. Along with that, a voice narrator explains each movement. The user can input the request sequence and starting head position and then select an algorithm to see it in action. The animation is shown step-by-step, which helps users visualize how the disk head travels between different cylinders. Apart from the visual animation, the project also calculates and displays performance metrics such as total seek time, average seek time, latency, and the number of head movements. Users can compare the performance of different algorithms using bar charts without having to input the data again.

2 OBJECTIVES

Objective 1: Simulation of Disk Scheduling Algorithms with Animation

The primary objective of this project is to simulate six standard disk scheduling algorithms: FCFS (First Come First Serve), SSTF (Shortest Seek Time First), SCAN, C-SCAN, LOOK, and C-LOOK. These algorithms are implemented using JavaScript and visually represented on an HTML5 Canvas. By animating the movement of the disk head as it processes a sequence of I/O requests, the simulation helps users better understand how each algorithm behaves in real-time. This visual approach makes it easier for learners to grasp the concepts behind disk scheduling by observing the path and pattern of the disk head movements.

Objective 2: Real-Time Voice Narration for Step-by-Step Explanation

To make the project more interactive and user-friendly, real-time voice narration is included using the Web Speech API. As the disk head moves from one cylinder to another, the system speaks out each step, such as "Moving from 53 to 98." This narration is synchronized with the animation, allowing users to hear and see the algorithm's operation simultaneously. This dual-mode (audio and visual) explanation makes it especially helpful for beginners and provides a better learning experience.

Objective 3: Performance Metrics Calculation and Display

Another key objective is to measure and display important performance metrics for each disk scheduling algorithm. These include total seek time, average seek time, latency (which is considered as half of the average seek time), and the total number of head movements. These metrics are essential to evaluate how efficiently an algorithm is handling disk I/O requests. After running an algorithm, the system displays all the values on the screen, making it easy to understand and compare the performance of different algorithms.

Objective 4: Storage of Results and Visual Comparison

The project also aims to store the results of each algorithm's execution using the browser's localStorage feature. This allows the system to remember performance data even after the browser is closed. When a user enters input once, the system runs all algorithms with the same data and stores the outputs. Then, a bar graph is generated to compare the total seek times of all algorithms visually. This comparison helps in quickly identifying which algorithm is most efficient for the given input and reduces the need for manually running each one separately.

3 METHODOLOGY

3.1 Overview

The project implements and visualizes several disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK using **JavaScript** and Plotly library for graphical representation.

Every algorithm is implemented as a distinct function that receives the array of disk requests, initial head location, direction, and max cylinder as arguments. A graph of head movement is returned along with a calculated seek time. Total seek time, average seek time, latency, and head movements are additional performance metrics calculated and saved locally using local Storage.

3.2 Input Specifications

The program accepts the following inputs from the user:

- **Request Queue**: Disk cylinder numbers that will be served.
- **Starting Head Position**: The initial position of the disk head.
- **Disk Size**: Largest size of a cylinder for the disk (for SCAN/C-SCAN algorithms).
- **Direction**: The direction of the first movement (Left or Right).

3.3 Execution of Algorithm

Every algorithm is a series of logical steps:

a) Request Sorting

For directional algorithms such as SCAN/C-SCAN, as well as LOOK/C-LOOK, the request array is ordered in ascending order. Ascending order makes it easy to traverse one way prior to wrapping/jumping back as necessary.

b) Simulation of Head Movement

Head movement is simulated through arrays y and x tracking X-coordinate (cylinder locations) and Y-coordinate (order/time) for plotting. Intermediate actions (such as jumping from one end to another in C-SCAN) are simulated through dashed lines (trace2) for differentiation.

c) Find Time Calculation

Seek time is calculated by the following formula:

seek += abs (current_position - next_request)

There is a cumulative sum kept for total seek time.

d) Performance Measures

The following calculations are made:

- **Total Seek Time**: Total distance traveled by the disk head.
- **Average Seek Time**: Average seek time is the seek time per request.
- **Latency**: About one-half of the mean seek time.
- **Head Movements**: Number of each individual movement from one cylinder to another, including jump movements for C-SCAN as well as for C-LOOK.

All the metrics are stored in local storage for comparison and audit.

3.4 Dealing with Edge Cases

Edge scenarios such as:

- Blank entry requests
- Head outside of the requested range
- Repeated requests within one location

are processed by conditional checks. In other instances, fallback handling utilizes SSTF as default handler for situations where the head is well out of range (as in clook() and cscan() routines).

3.5 Visualization

Using the Plotly.js library, the head movement is represented as:

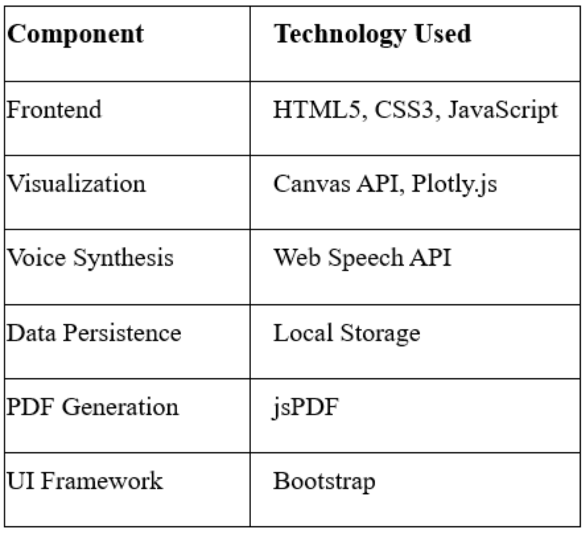
- Normal movement between requests with solid line
- Dashed line for jumps (C-SCAN, C-LOOK)
- Directional time-plots by decreasing time for clarity

4 SYSTEM ARCHITECTURE

The Disk Scheduling Visualization System follows a client-side web architecture with:

1. **Frontend:** HTML/CSS/JavaScript with Plotly.js for visualization
2. **Data Storage:** Browser's local Storage for performance metrics
3. **Animation:** Canvas API and Web Speech API

Table.1.Technical Stack



Component	**Technology Used**
Frontend	HTML5, CSS3, JavaScript
Visualization	Canvas API, Plotly.js
Voice Synthesis	Web Speech API
Data Persistence	Local Storage
PDF Generation	jsPDF
UI Framework	Bootstrap

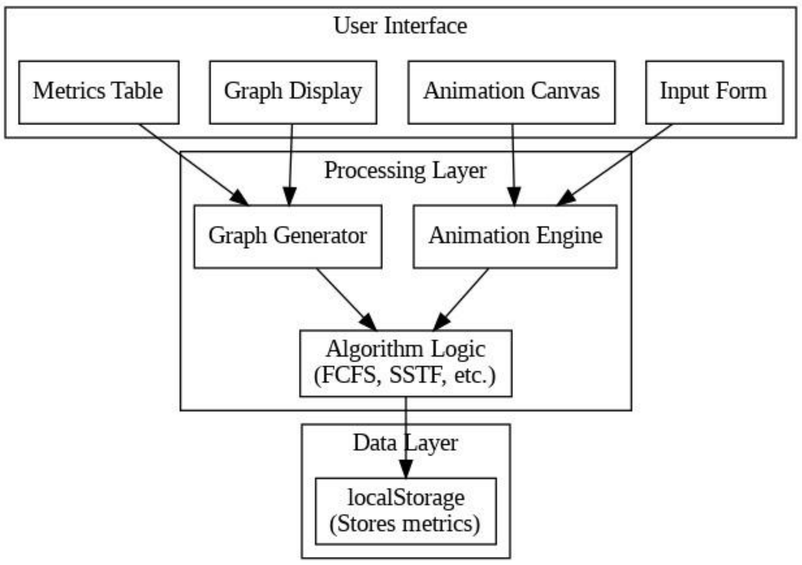


Fig.1. System design block diagram

5 IMPLEMENTATION

The Disk Scheduling Visualization Tool is developed using front-end web technologies such as HTML, CSS, Bootstrap, and JavaScript.

Technologies Used:

- **HTML5 & Bootstrap:** For building the structure and responsive design of the web page.
- **JavaScript:** To handle algorithm logic, canvas animation, and voice narration.
- **Canvas API:** To draw visual animations for disk head movements.
- **Web Speech API:** To implement the voice narration feature.

Working Process:

1. **Input Collection:**

The user enters the starting head position, request queue, and selects a disk scheduling algorithm. Once submitted, the values are passed to the JavaScript logic.

2. **Algorithm Execution:**

The selected algorithm is executed, and it calculates the order in which the disk head should move to serve the requests. The seek sequence is stored in an array.

3. **Animation:**

Using the Canvas API, the disk head starts moving in animated steps from one request to another. Circles represent the requests, and arrows indicate the head's movement path.

4. **Voice Narration:**

Just before each movement step, the Web Speech API announces the transition like "Moving from 53 to 183." In the animation, the voice narration and the movement of the disk head happen together making it easy to follow and understand.

5. **Seek Time and Output Display:**

After all movements are completed, the seek sequence is displayed along with the total seek time. A chart comparing the performance of different algorithms is also shown on the performance page.

6. The **Comparison Module** runs all the algorithms automatically when the user gives input once. After that, it shows a bar graph comparing the total seek times of all algorithms.

7. **Performance Metrics:**

In this project, we calculate how far the disk head moves (total seek time), and then find the average by dividing it by the number of requests. We also find latency, which is half of the average seek time, and count how many times the head moves. These numbers help us understand how efficient each algorithm is.

8. **Using local Storage:**

We store all these results in the browser using local Storage. This means the data stays saved even if the page is closed. Users can compare different algorithm runs easily, and no internet is needed. It helps in tracking which algorithm works best for the given input.

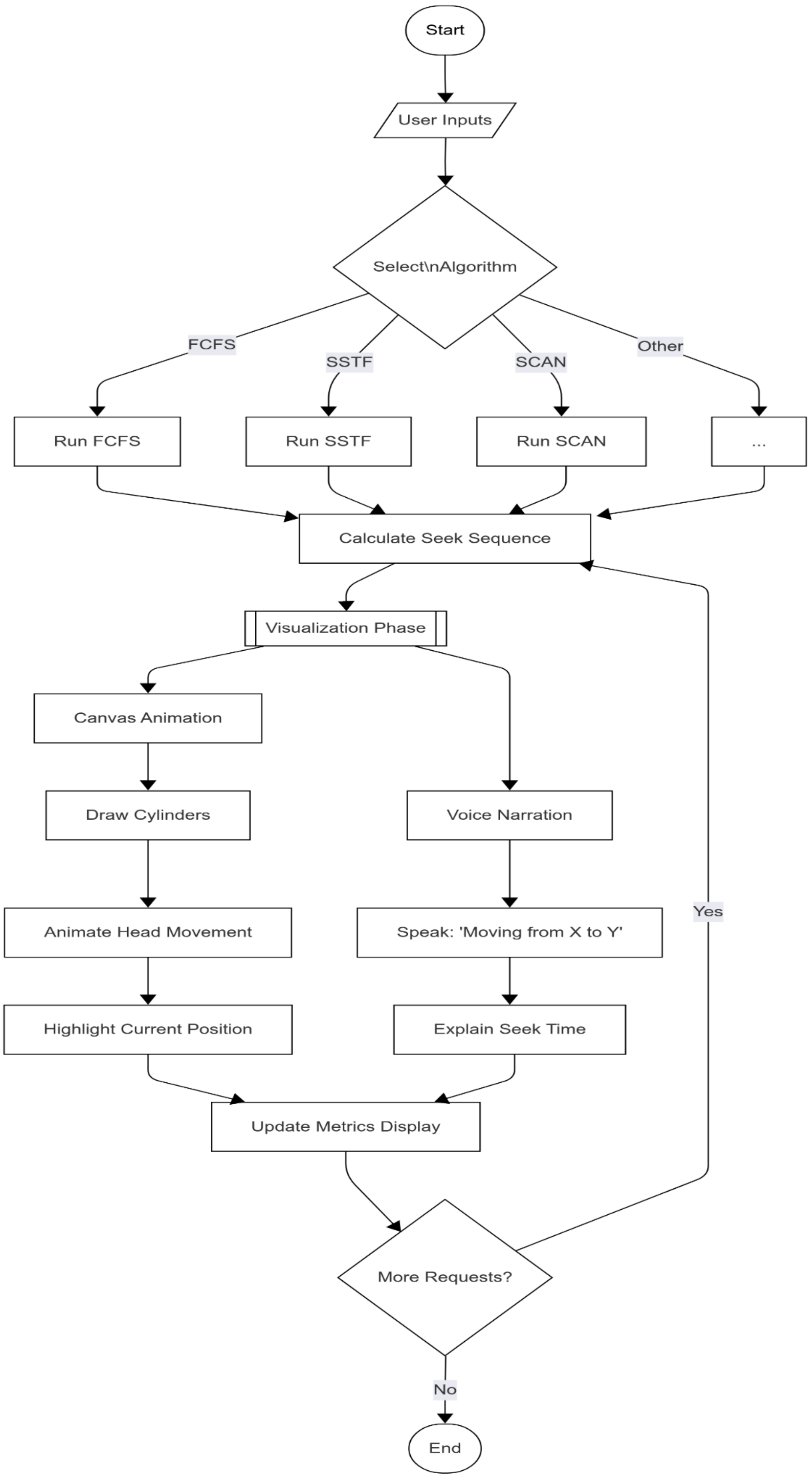
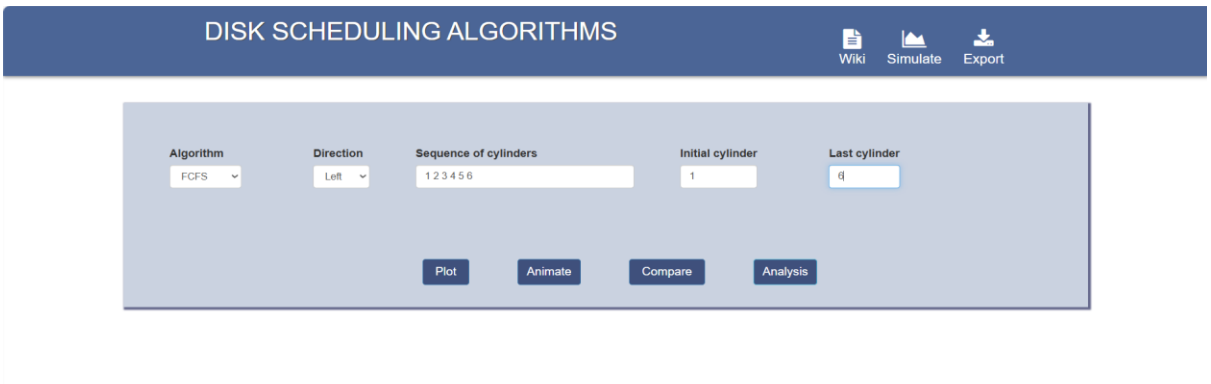


Fig.2.Flow Diagram

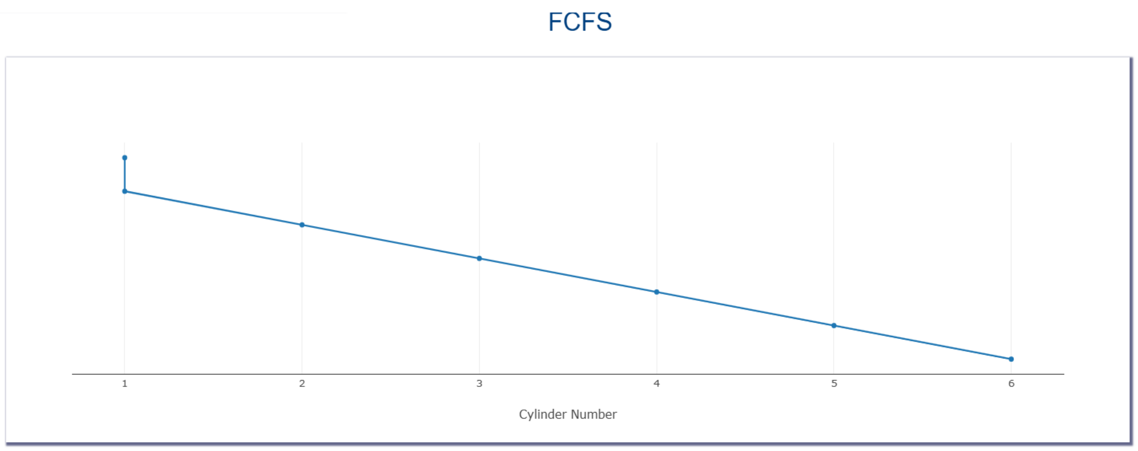
6 RESULTS

- The system correctly runs all six disk scheduling algorithms: FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK.
- It shows the movement of the disk head using animation on the canvas, making it easy to understand.
- The voice says the current disk head movement, and it matches perfectly with the animation.
- The system calculates total seek time, average seek time, latency, and head movements for every algorithm.
- All these results are saved in the browser using localStorage, so users can see them later too.
- It shows a bar graph to compare all algorithms based on seek time using the same input.
- This tool is better than other existing ones because it includes animation, voice, graphs, and result saving—all in one place.



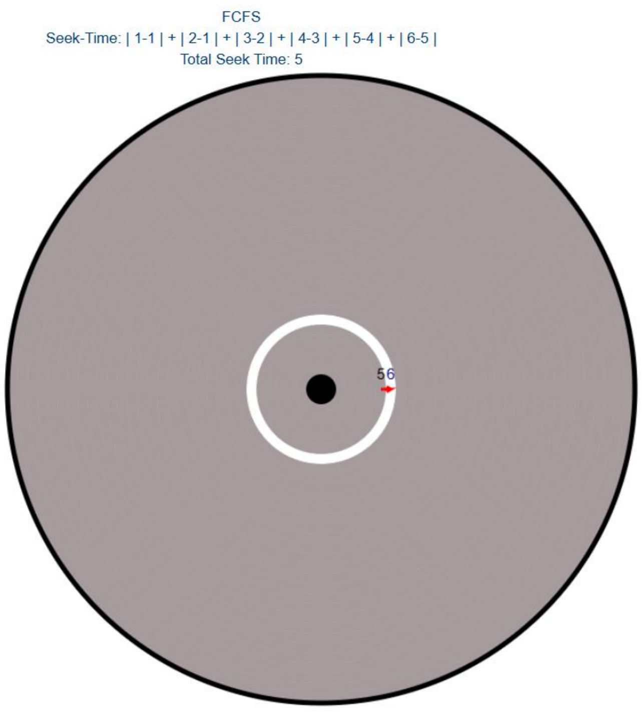
Algorithm	Direction	Sequence of cylinders	Initial cylinder	Last cylinder
FCFS	Left	1 2 3 4 5 6	1	6

Fig.3.User Page



Seek-Time: |1-1|+|2-1|+|3-2|+|4-3|+|5-4|+|6-5| = 5

Fig.4.Disk Movement Graph


FCFS

Seek-Time: | 1-1 | + | 2-1 | + | 3-2 | + | 4-3 | + | 5-4 | + | 6-5 |

Total Seek Time: 5

Fig.5.Disk Head Movement

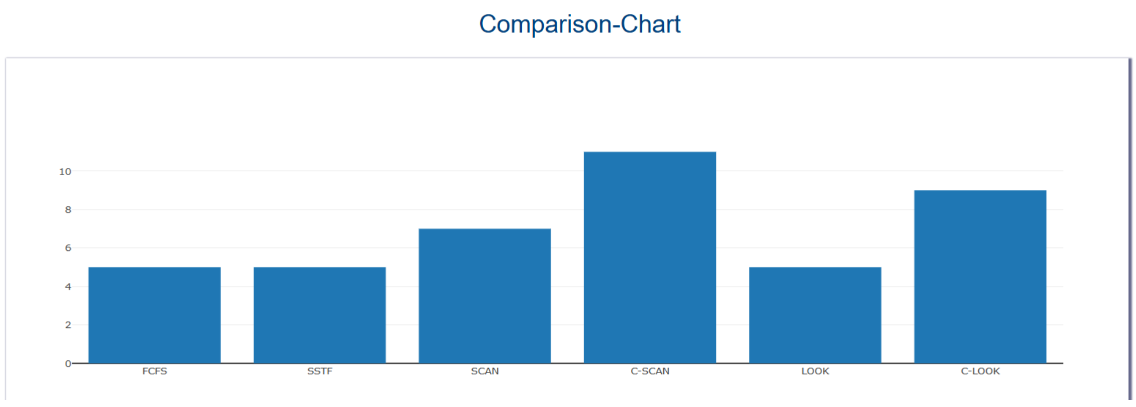
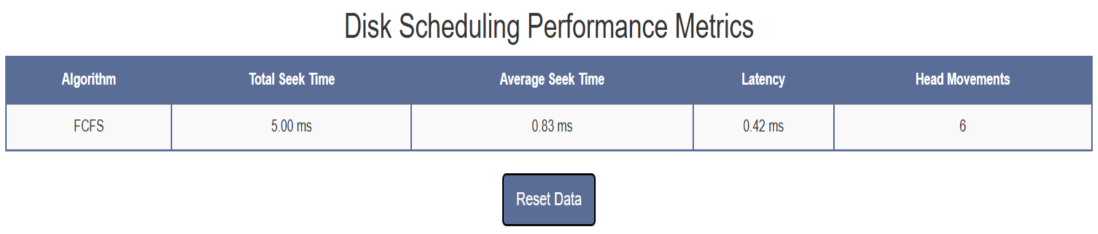


Fig.6.Comparison Graph


Disk Scheduling Performance Metrics

Algorithm	Total Seek Time	Average Seek Time	Latency	Head Movements
FCFS	5.00 ms	0.83 ms	0.42 ms	6

Fig.7.Analysis for the Algorithm

7 FUTURE WORKS

1. Support for Real-Time Disk Scheduling:

Can simulate real-time scenarios where disk requests arrive dynamically during the animation, instead of all being given at once.

2. Add More Scheduling Algorithms:

Additional algorithms like N-Step SCAN or FSCAN can be added to the simulator for more comprehensive comparisons and learning.

3. Dark Mode and UI Improvements:

A user-friendly dark mode and better responsive design can be implemented to improve the visual experience on both desktop and mobile devices.

4. Multi-language Voice Narration:

To support a wider range of students, the voice explanation feature can be extended to multiple languages, making it more accessible.

5. AI-Based Algorithm Recommendation:

AI can be integrated into the system to automatically suggest the most efficient disk scheduling algorithm based on the user's input pattern. This would help beginners choose the best option without needing to analyze all metrics manually.

8 REFERENCES

[1] Galvin, Silberschatz, Gagne – “Operating System Concepts”, Wiley

[2] Mozilla Web Docs – Web Speech API

[3] HTML5 Canvas Tutorials – W3Schools

[4] Plotly.js Documentation – https://plotly.com/javascript/

[5] Bootstrap Documentation – https://getbootstrap.com/

[6] JavaScript Documentation – https://developer.mozilla.org/en-US/docs/Web/JavaScript