**State University of New York at New Paltz**

**Matthew Morfea**

**Project Type: On-Campus Project Assignment, Class Key: c-s21-20, CPS485-01**

**OS Visualizations (Group No. 6)**

**FINAL PROJECT REPORT**

**LINK:** [**http://cs.newpaltz.edu/p/s21-06/**](http://cs.newpaltz.edu/p/s21-06/)

**LOGIN: morfeam1**

**Computer Science Projects**

**Spring 2021**

**(Prof. Hanh Pham)**

**TABLE OF CONTENTS**

**1. Problem Description**

**1.1 Business Context and Goals** ….……………………………………….…………….……. page 03

**1.2 Technical Requirements** ….……………………………………………….……….………. page 04

**1.3 Your Responsibilities** ……...……………………………………………….……….………. page 04

**2. Technologies**

**2.1 Related Technologies** ….……………………………………….……………………..…….. page 05

**2.2 Newly Learned Skills/Technologies …**………………………………..………………. page 05

**3. Design**

**3.1 System Architecture** ………………….……………………………………….………………. page 06

**3.2 Components** ………………………….…………………………………………..………………. page 08

**4. Software/System Description**

**4.1 Map of Files** ………………………….…………………………………………..………………. page 10

**4.2 DATA Format & Description** ………………………………………………………………. page 11

**4.3 Developer’s Guide** …..…………….…………………………………………..………………. page 11

**4.4 User’s Guide** …..…………….…………….……………………………………..………………. page 12

**5. Test Results (Functionality/Performance)**

**5.1 Test Function 1/Transaction # 1** …………………………..….…….…..………..……. page 14

**5.2 Test Function 2/Transaction # 2** …………………………..….…….……..……..……. page 15

**5.3 Test Function 3/Transaction # 3** …………………………..….…….…..………..……. page 16

**5.4 Test Function 4/Transaction # 4** …………………………..….…….……..……..……. page 17

**5.5 Test Function 5/Transaction # 5** …………………………..….…….…..………..……. page 18

**5.6 Test Function 6/Transaction # 6** …………………………..….…….……..……..……. page 18

**5.7 Test Function 7/Transaction # 7** …………………………..….…….…..………..……. page 19

**6. Professional and Career Benefits**

**6.1 Challenges and Solutions** ……….…….……..……………………………………….……. page 22

**6.2 Other Learned Lessons** …….…….…….……..……………………………………….……. page 24

**7. Conclusions** ………………………….……………………………………………...…………..………………. page 25

**8. References** ………………………….…………………..………………………………………..………………. page 26

**9. Appendixes** …………………….………………….……………………………………………..………………. page 27

**1. Problem Description**

**1.1 Business Context and Goals**

Sometimes it’s hard to learn about algorithms based on concepts or equations, especially for a huge and complex field like Operating Systems. This project is designed to focus on the visualization of algorithms pertaining to different concepts in Operating Systems. The software we created was designed to aid teachers and students demonstrate and learn the varying algorithms of Operating Systems. This includes many algorithms from different places of the wide field of Operating Systems. This includes areas such as CPU Scheduling, Disk Scheduling, Memory Allocation, Page Replacement Methods, File Allocation, and others. Being attached to the SUNY New Paltz web server, Users (teachers, students, etc.) can use this software wherever they can connect to the internet. This could be in a classroom giving a demonstration, in a dorm or home studying, or simply on a mobile device to dive into new knowledge in the world of Operating Systems.

Our software is hosted on the SUNY New Paltz web server and is accessible to anyone with an internet connection. This will connect people to the front-end website where they can learn and interact with the algorithms and visualizations provided on the web server. Each algorithm has buttons and features based on which class of algorithms you would be looking at. Some of the buttons include Previous and Next features allowing you to move one step at a time to show the logic behind the algorithm. Play and Pause, which show a slowed down version of the algorithm along with highlights of what specifically the algorithm is changing as it progresses through the input and outputs in real time. Start and end, that simply display before and after the algorithm has run its course through the many input and output files.

Connections to other files and File I/O on the SUNY New Paltz Web server is taking place in the background to help generate and connect the input and output files to the front-end website as well as the back-end algorithms. Displays of these buttons as well as connections to the input/output files is shown through the front-end website, which was created using common web design techniques.

Overall, the software’s core is based on algorithm implementation and the Graphics User Interface. The connection to the web server allows the input and output files to connect to the front-end side of the website. This allows users to input button clicks to view certain steps of the algorithm which are then outputted based on the button clicks. Based on the information provided the user can learn more about the Algorithms of Operating Systems.

**1.2 Technical Requirements**

The technical requirements are straightforward for our overall design of the project. The software can be viewed through the SUNY New Paltz web server, thus, anywhere with an internet connection and search engine such as google, Firefox, etc., its hosted on a website. The user makes clicks via mouse and keyboard (or touchscreen) to interact with the main components of the website. The main components of the website being the front-end of the website itself and the actual algorithm’s visualizations in the form of graphics user interfaces. Any standard computer of smart phone would be able to access both the website and the graphics user interface.

The processing between the website to user and algorithm to user is as follows:

1. User connects to website to display home screen.
2. User clicks a specific algorithm, then input data is displayed for that algorithm via the graphics user interface.
3. As the user determines which buttons to be pressed the algorithms processes the user’s information accordingly:
   1. Play: Algorithms outputs all numbers / information in sequence of steps.
   2. Next: Algorithm outputs one number / part of information.
   3. Previous: GUI outputs previous Processes numbers / information.
   4. Pause: Pauses play feature wherever in the processing.
   5. Start: Goes to start before algorithms output.
   6. End: Goes to end after algorithms output.
   7. Reset: Reset’s algorithm and algorithm choice to the beginning values.
4. The user can then move to another class of algorithms or another algorithm through navigation on the website and buttons.

**1.3 Your Responsibilities**

My responsibilities are specifically for the Disk Scheduling section of algorithms. This includes many different algorithms like FCFS (First Come First Serve), SSTF (Shortest Seek Time First), CSCAN (Circular SCAN), LOOK, CLOOK (Circular LOOK), and others. I was required to implement these algorithms as well as create a smooth and straight forward visualization in the form of a graphics user interface for each. I implemented the data for these algorithms directly in text files for each algorithm known on the web server as input.txt. These text files where then run through Java code also placed on the web server in order to sort the read / write heads correctly as to how they would be processes by the different algorithms. Then this information in passed to each algorithm output.txt file respectively. These files are then read on the front-end website’s side to be sent to arrays in JavaScript. These arrays are sent to JavaScript code that draw lines that represent the paths that the read / write head would take on an actual disk drive. I am responsible for these concepts and the display of the visualization of all Disk Scheduling algorithms and the web design of the Disk Scheduling Graphics User Interface (GUI).

**2. Technologies**

**2.1 Related Technologies**

For the software and my contributions to the project, I am using a combination of many front-end languages on our web server. I am using HTML5, CSS3, and Bootstrap5 to design portions of the web design to house the algorithms. Furthermore, I am using HTML5, CSS3, JavaScript (HTML) DOM, JavaScript, and HTML Canvas to create the visualizations for the algorithms mentioned in section 1.3. This is all done by hosting the web server through SUNY New Paltz’s Wyvern Server, which I am accessing through WinSCP. All the code being typed up in Visual Studio Code. For the design portions, I am using the Microsoft Office Suite to create flowcharts, PowerPoints, and type up reports.

For local testing on my own server, I used XAMPP, which is a local host simulated server, to simulate and test File I/O passing in a server-based setting.

**2.2 Newly Learned Skills/Technologies**

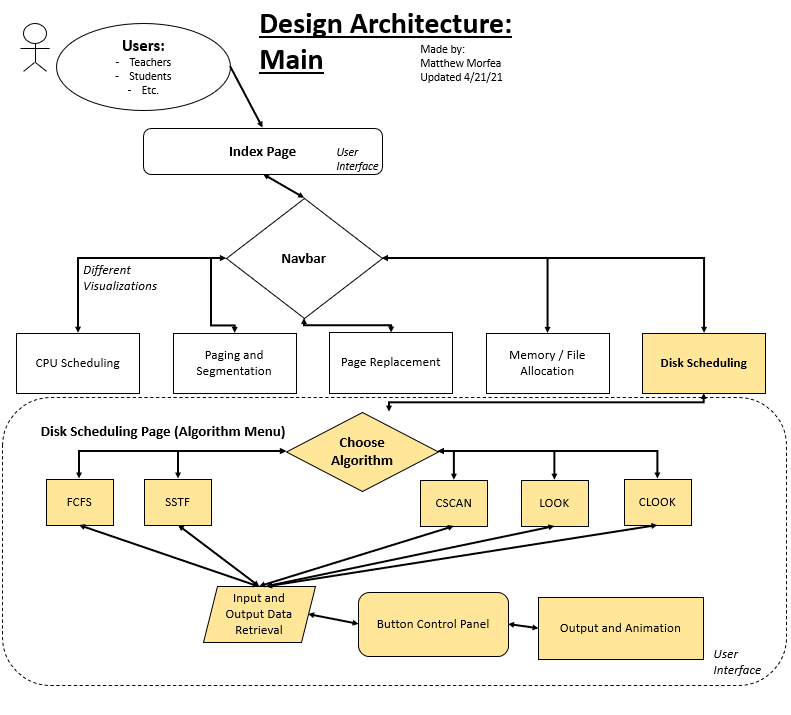
There are many new skills and technologies that I worked on through the creation of this project. Some of which being:

* JavaScript
* JavaScript (HTML) DOM
* HTML CANVAS
* XAMPP
* Putty
* Front-end File I/O Techniques
* PHP
* AJAX

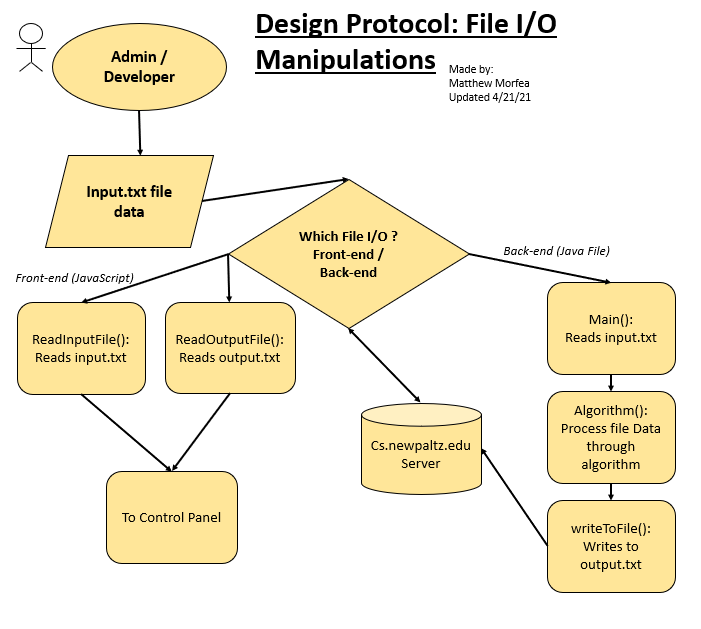
**3. Design**

**3.1 Ideas for Solution (Architecture + Protocols)**

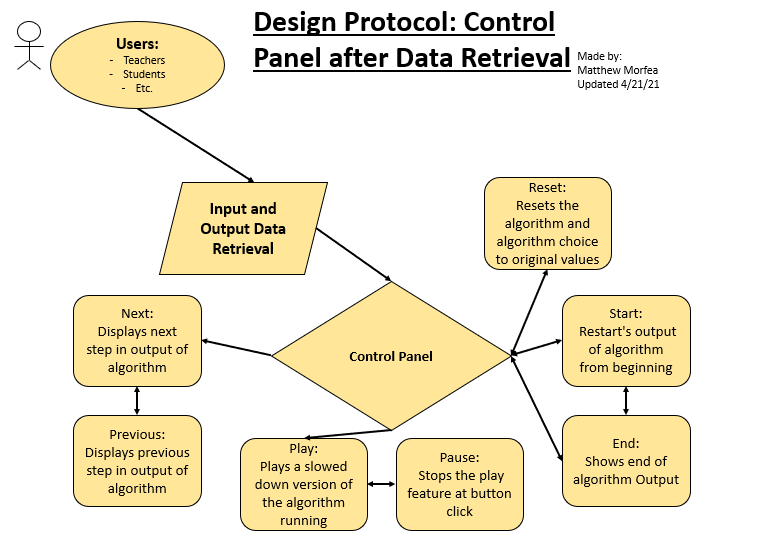
Architecture: Main Program



Protocol: File I/O Manipulations (Back-end and Front-end)



Protocol: (Buttons) Control Panel

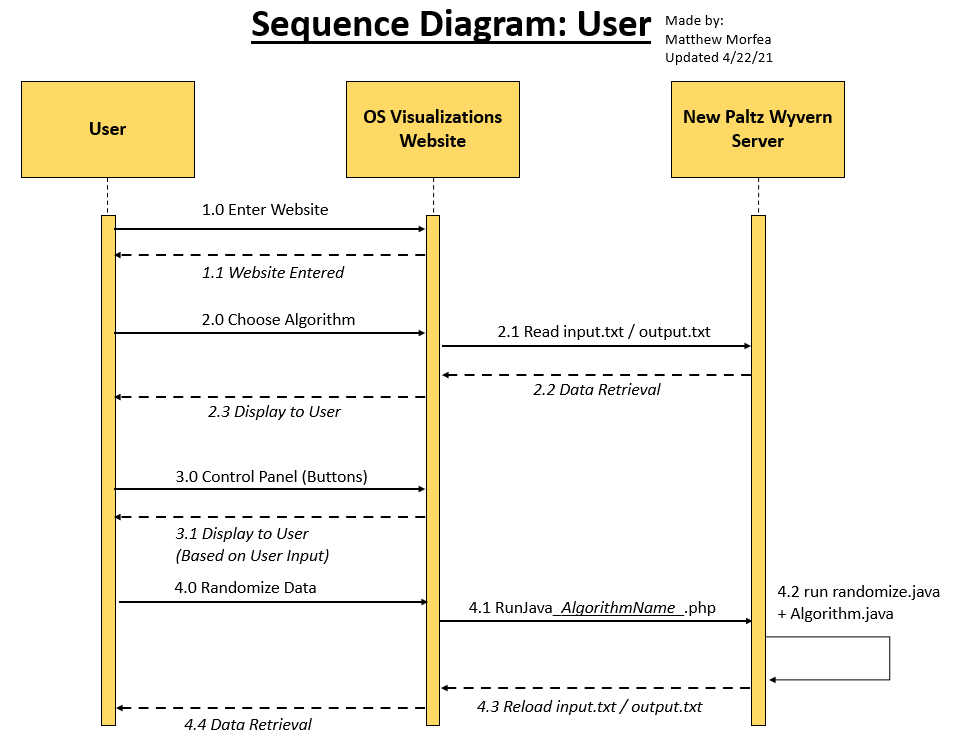


**3.2 Programming/Coding Components**

For this project I used this list of Languages, Packages and Tools:

* HTML5
* HTML Canvas
* CSS3
* Bootstrap5
* Standard JavaScript
* JavaScript DOM
* WinSCP
* Visual Studio Code
* XAMPP
* PHP
* AJAX
* Java (via BlueJ)
* Text File I/O Manipulation Techniques

Sequence Diagram: Processing Through Components



**4. Software / System Description**

**4.1 Map of Files**

Map of Files for Disk Scheduling portions:

:

New Paltz CS Wyvern Server…

:

/s21-v4 /  
 |

|--- /01-cpu/ *(Group Member)*

|--- /02-replace/ *(Group Member)*

|--- /03-disk/

|--- “Capture.JPG”

|--- “Capture2.JPG”

|--- “DiskSchedulingV5.html”  
     |---/Older Versions/ … *(Folder of Older / Unused Material)*

|---/Algorithms/

|

|---/FCFS/  
               |--- "input.txt"  
               |--- "output.txt"  
               |--- "FCFS.java/class"

|--- "randomize.java/class"

|--- "JavaRunFCFS.php"

|---/SSTF/  
               |--- "input.txt"  
               |--- "output.txt"  
               |--- "SSTF.java/class"

|--- "randomize.java/class"

|--- "JavaRunSSTF.php"

|---/CSCAN/  
               |--- "input.txt"  
               |--- "output.txt"  
               |--- "CSCAN.java/class"

|--- "randomize.java/class"

|--- "JavaRunCSCAN.php"

|---/LOOK/  
               |--- "input.txt"  
               |--- "output.txt"  
               |--- "LOOK.java/class"

|--- "randomize.java/class"

|--- "JavaRunLOOK.php"

|---/CLOOK/  
               |--- "input.txt"  
               |--- "output.txt"  
               |--- "CLOOK.java/class"

|--- "randomize.java/class"

|--- "JavaRunCLOOK.php"

|--- /04-memory/ *(Group Member)*

|--- /05-files/ *(Group Member)*

|--- /06-address/ *(Group Member)*

|--- “index.html”

**4.2 Data Format**

Input.txt and Output.txt are laid out as follows:

**Input.txt / Output.txt**

*0 699*: - Size of HTML Canvas Window on Website

*50:* - Head = Start of Read / Write

*102 402 505 25 32*: - Queue = Different Read / Write Requests

**4.3 Developer’s Guide**

**Guide A.** Access and Edits in Live Host (Recommended WinSCP):

Step 1:

Access the location where the website / files are being hosted.

* + Currently this is the SUNY New Paltz Wyvern Server located at cs.newpaltz.edu.
  + Login to website through WinSCP and locate ../s21-06/ files.
  + *Note*: This can be done through other servers if all files in ../s21-06/ are downloaded and file connections changed within code where needed.

Step 2:

Once you are in the file location, you may enter different OS Visualizations worked on by different developers.

* + This includes CPU Scheduling, Page Replacement, Disk Scheduling, Memory Allocation, File Allocation, and Page and Segmentation Addressing.
  + You may also access the central web framework located at index.html within ../s21-06/s21-v4/

Step 3:

Using WinSCP you can open and close files directly through their notepad IDE format and relaunch these portions of the code by saving on the hosted live server.

* + *Note*: Working in WinSCP’s notepad IDE (Integrated Development Environment) is tedious and is only recommended for small changes on live server. The best way to right code is through proper IDE on a local host.

Step 4:

Adding and removing files is easy as you can drag and drop files from your computer, and they are copied onto the live server via the WinSCP interface.

**Guide B.** Access and Edits in Local Host (Recommended XAMPP):

Step 1:

Its important to download and setup XAMPP properly on your device, do this through the instructions on XAMPP’s website under download.

Step 2:

Move your files into htdocs of XAMPP and run ***apache*** server through XAMPP’s GUI, in order to start your local host.

* + These files can now be accessed through different IDEs (Ex: Visual Studio Code / BlueJ) for a better formatted and easier to read coding experience.
  + These local files can be added to the Live server by Steps 3 and 4 through Guide A.

**\*\*\* Developers have access to all what Users can do. All the user’s features are included in developers features. \*\*\***

**``` For more information on Development please see Appendices A and B. ```**

**4.3 User’s Guide**

**Guide A.** Accessing the website:

Step 1:

Open a web browsers and type in live websites URL (<http://cs.newpaltz.edu/p/s21-06/s21-v4/>).

Step 2:

Choose a Visualization that you would like to view.

* + Options include CPU Scheduling, Page Replacement, Disk Scheduling, Memory Allocation, File Allocation, and Page and Segmentation Addressing.
  + This is available through the navigation bar of menu of the homepage (index.html).

Step 3:

*If Disk Scheduling is chosen*, choose an algorithm to view (via buttons).

* + This includes FCFS, SSTF, CSCAN, LOOK, and CLOOK.
  + You may also choose to randomize results. This changes the values in the input.txt and output.txt to see different datasets.
  + *Note*: Other pages were made by other developers. Guides may vary.

Step 4:

Use control panel (buttons) to view the different movements of the read / write head on the graph down below. Button Features:

* + Next: Displays next step in output of algorithm
  + Previous: Displays previous step in output of algorithm
  + Play: Plays an incremental version of the algorithm running
  + Pause: Stops the play feature at button click
  + Start: Restart's output of algorithm from beginning
  + End: Shows end of algorithm Output
  + Reset: Resets the algorithm and algorithm choice to original values

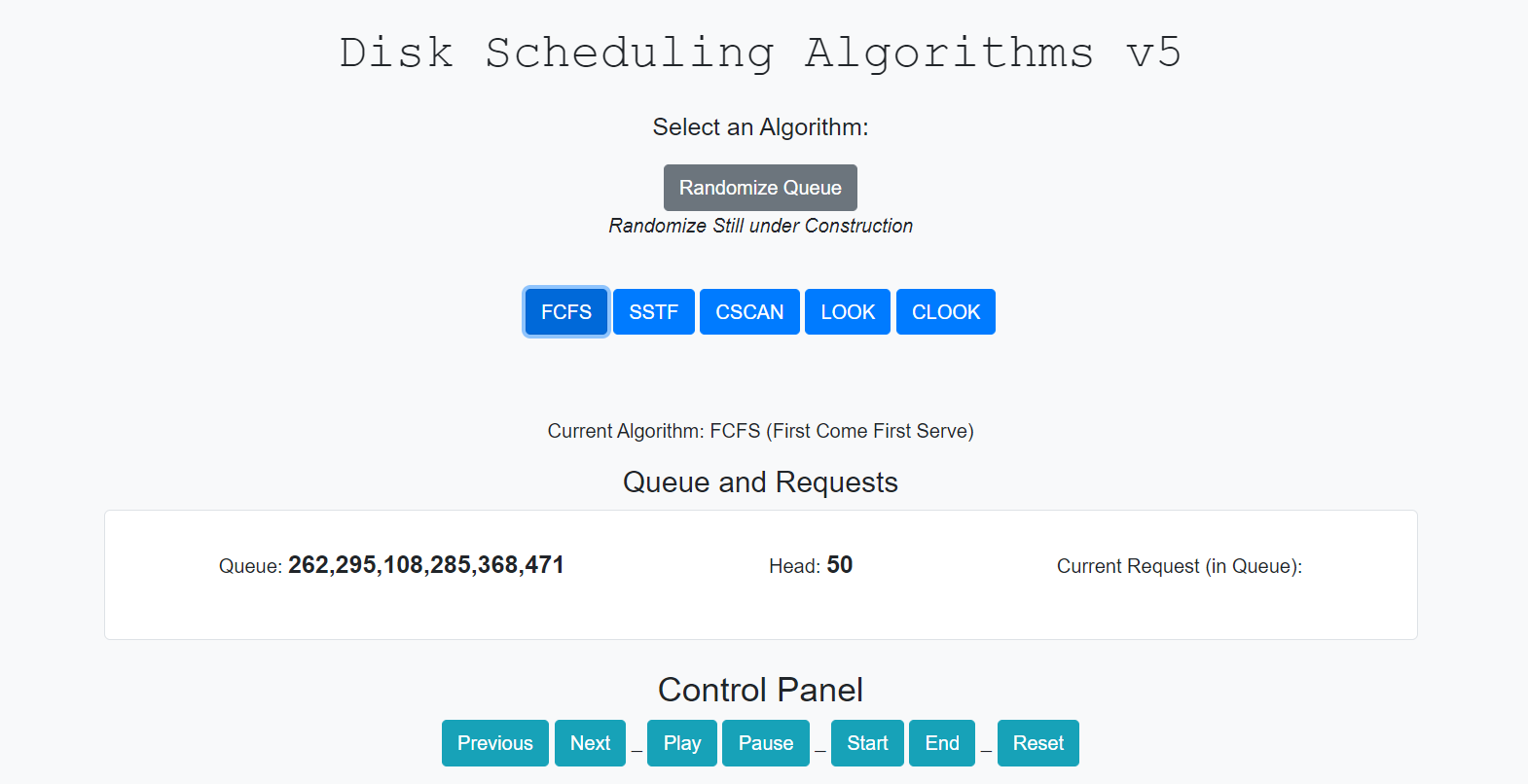
This control panel will allow you to view the different traces of the algorithm. The values of the queue and read / write head are located above the graph.

**5. Test Results (Functionality / Performance)**

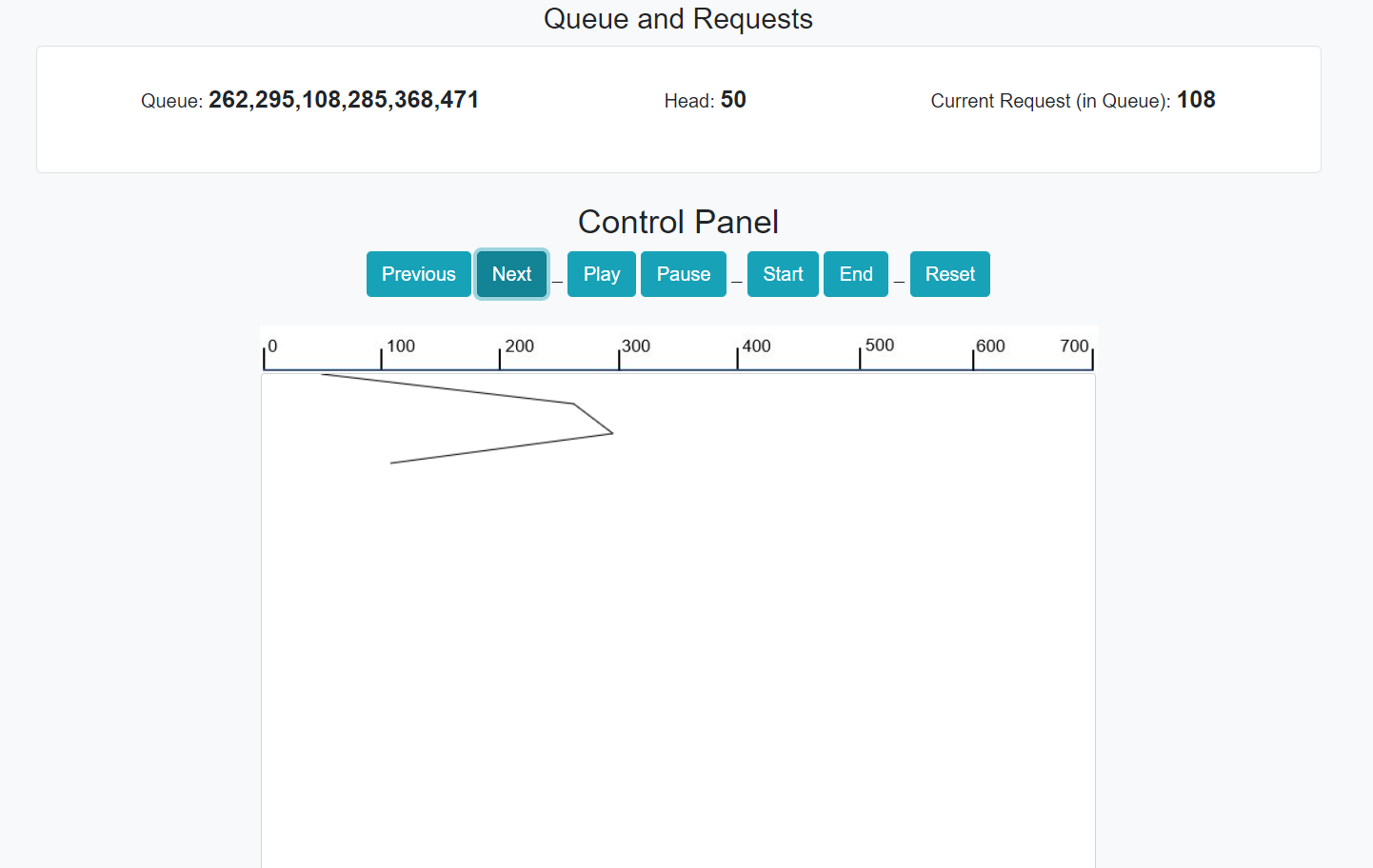
***\*\*Sidenote of basic website layout down below for reference / control.***

**5.1 Test Function 1: FCFS**

Disk Scheduling FCFS Algorithm (First Come First Serve):



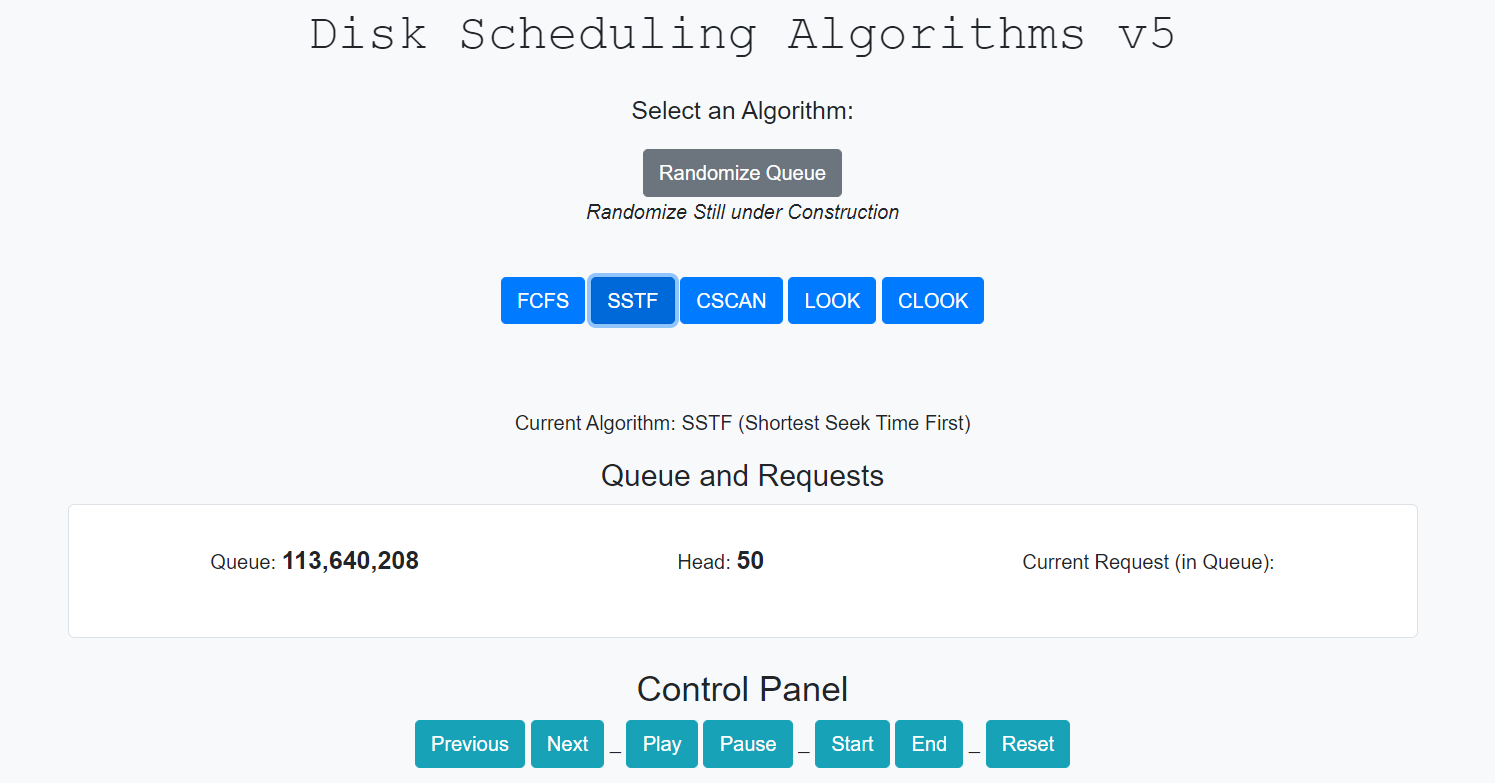
* Upon button click, FCFS’s input.txt and output.txt are retrieved.
* Queue and Head from input.txt are displayed, Current Request is awaiting user input to control panel.



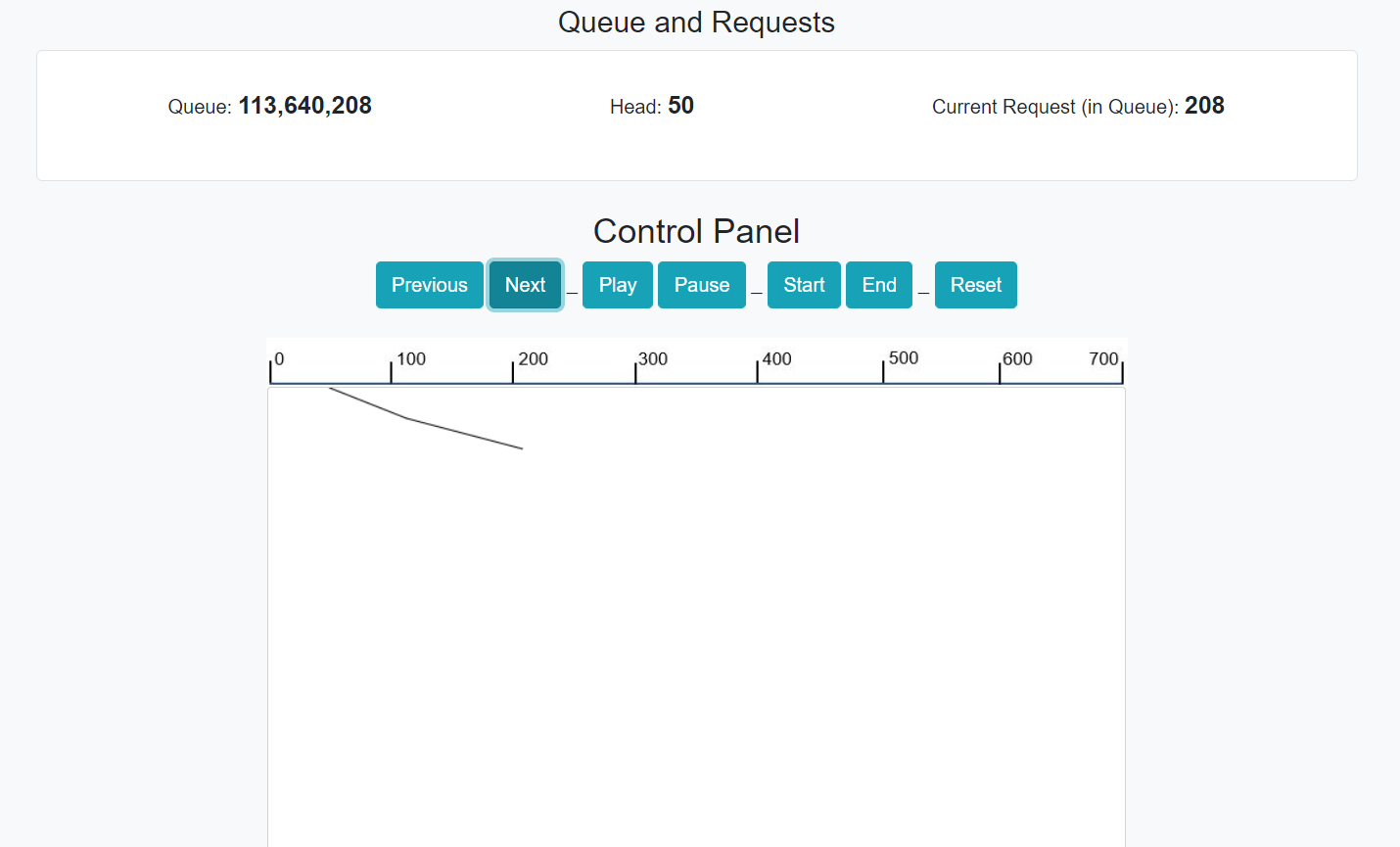
* Upon Control Panel, next button click was clicked 3 times, displaying the proper location in queue for FCFS, in this example this is 108, according to output.txt.
* All other buttons on Control Panel worked as expected for FCFS algorithm.

**5.2 Test Function 2: SSTF**

Disk Scheduling SSTF Algorithm (Shortest Seek Time First):



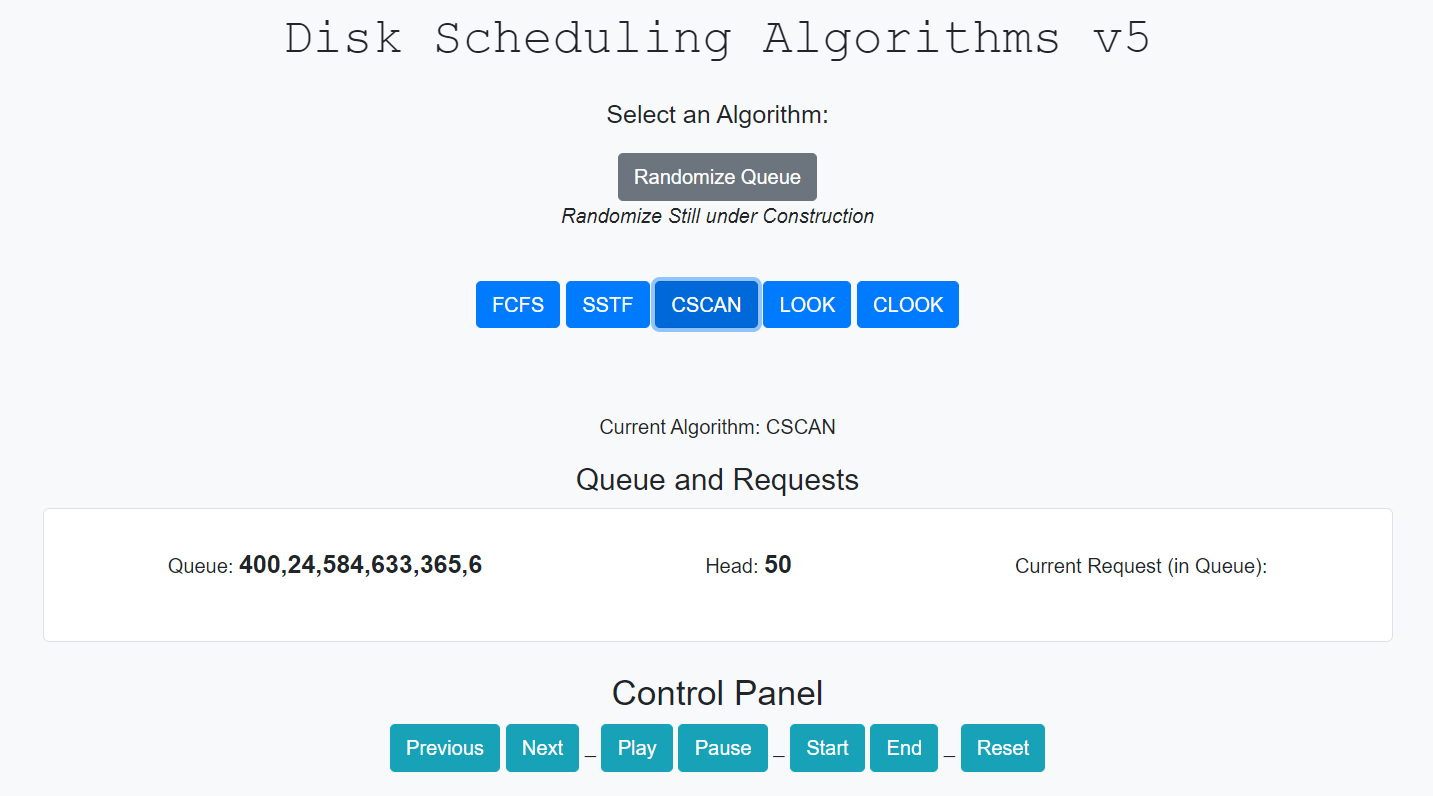
* Upon button click, SSTF’s input.txt and output.txt are retrieved.
* Queue and Head from input.txt are displayed, Current Request is awaiting user input to control panel.



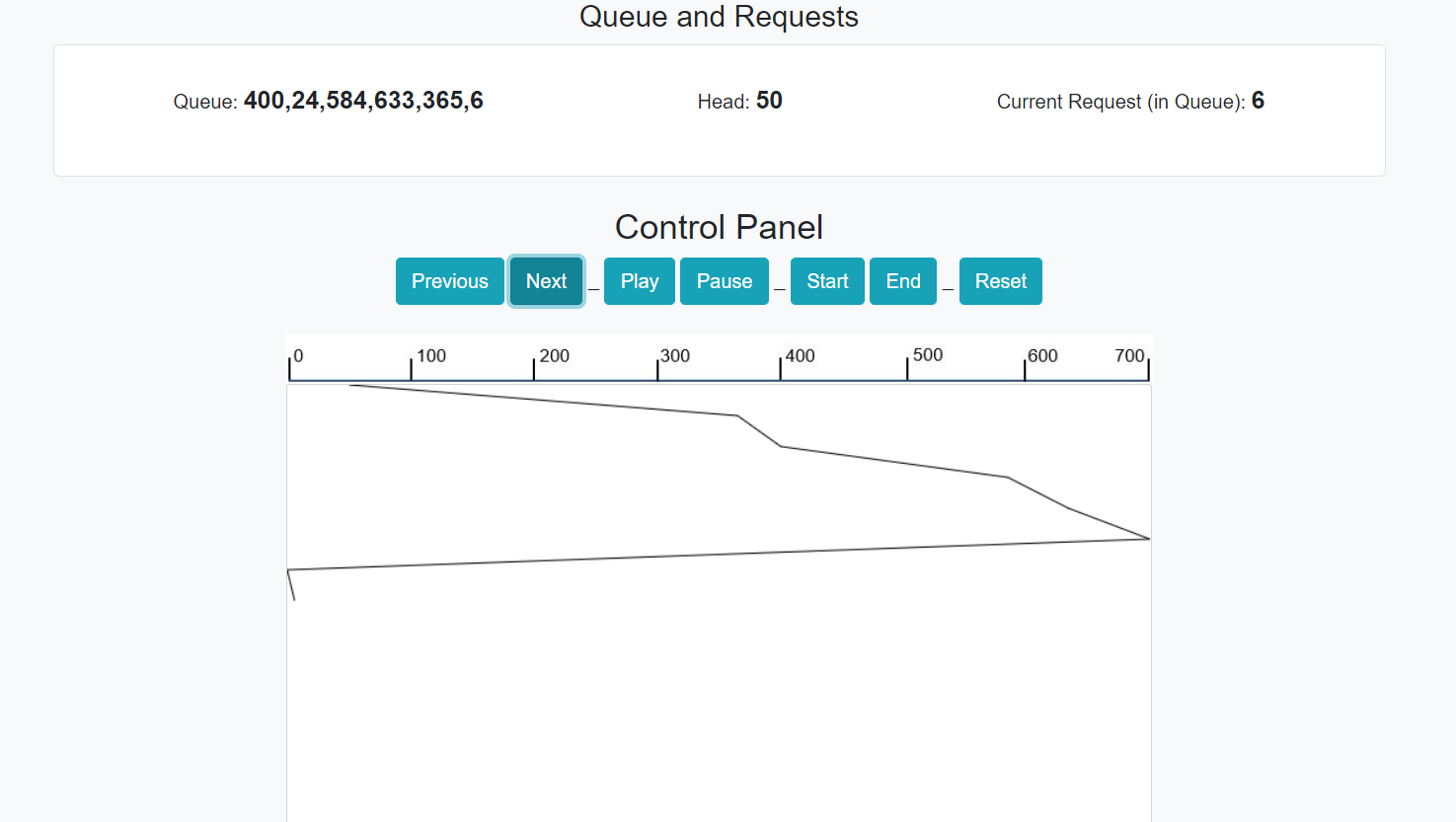
* Upon Control Panel, next button click was clicked 2 times, displaying the proper location in queue for SSTF, in this example this is 208, according to output.txt.
* All other buttons on Control Panel worked as expected for SSTF algorithm.

**5.3 Test Function 3: CSCAN**

Disk Scheduling CSCAN Algorithm:



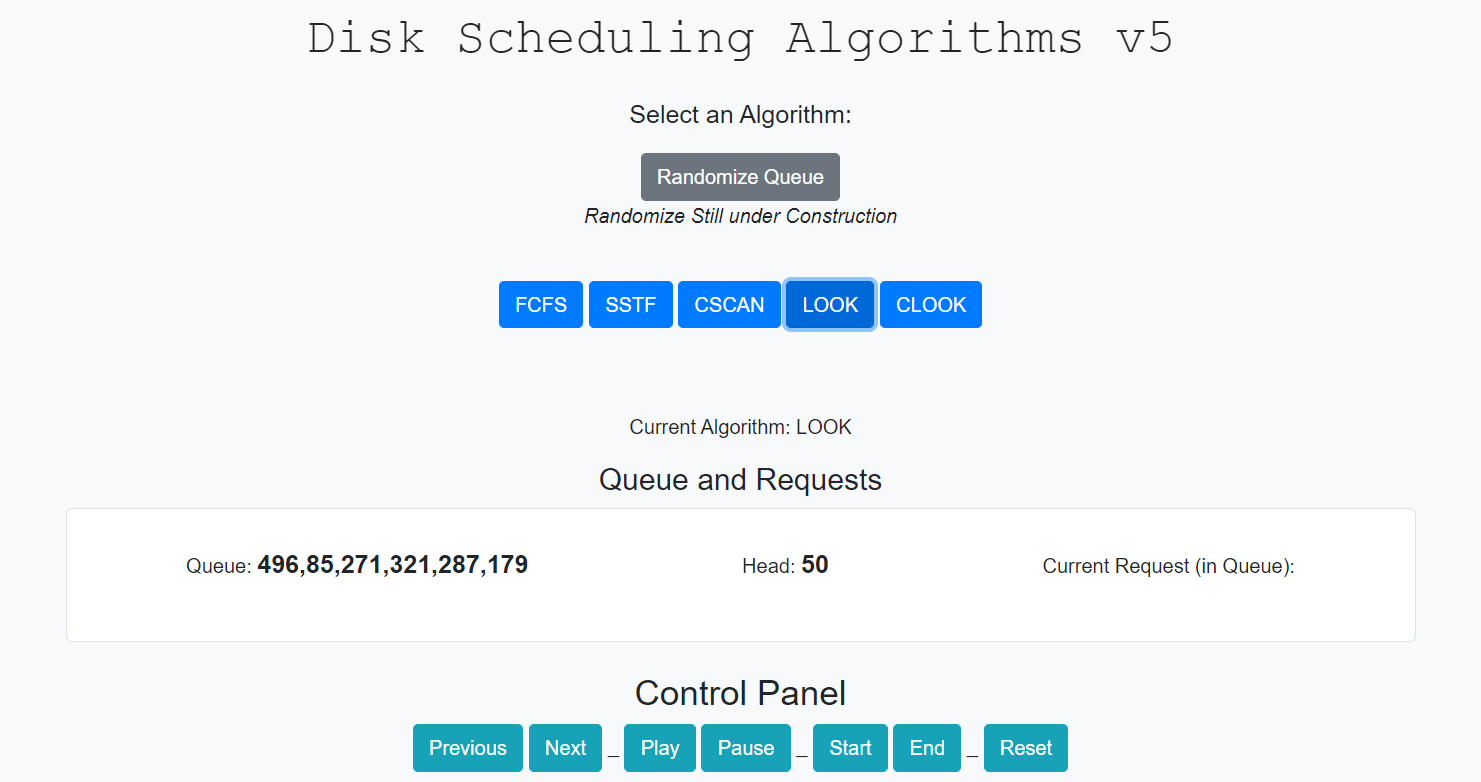
* Upon button click, CSCAN input.txt and output.txt are retrieved.
* Queue and Head from input.txt are displayed, Current Request is awaiting user input to control panel.



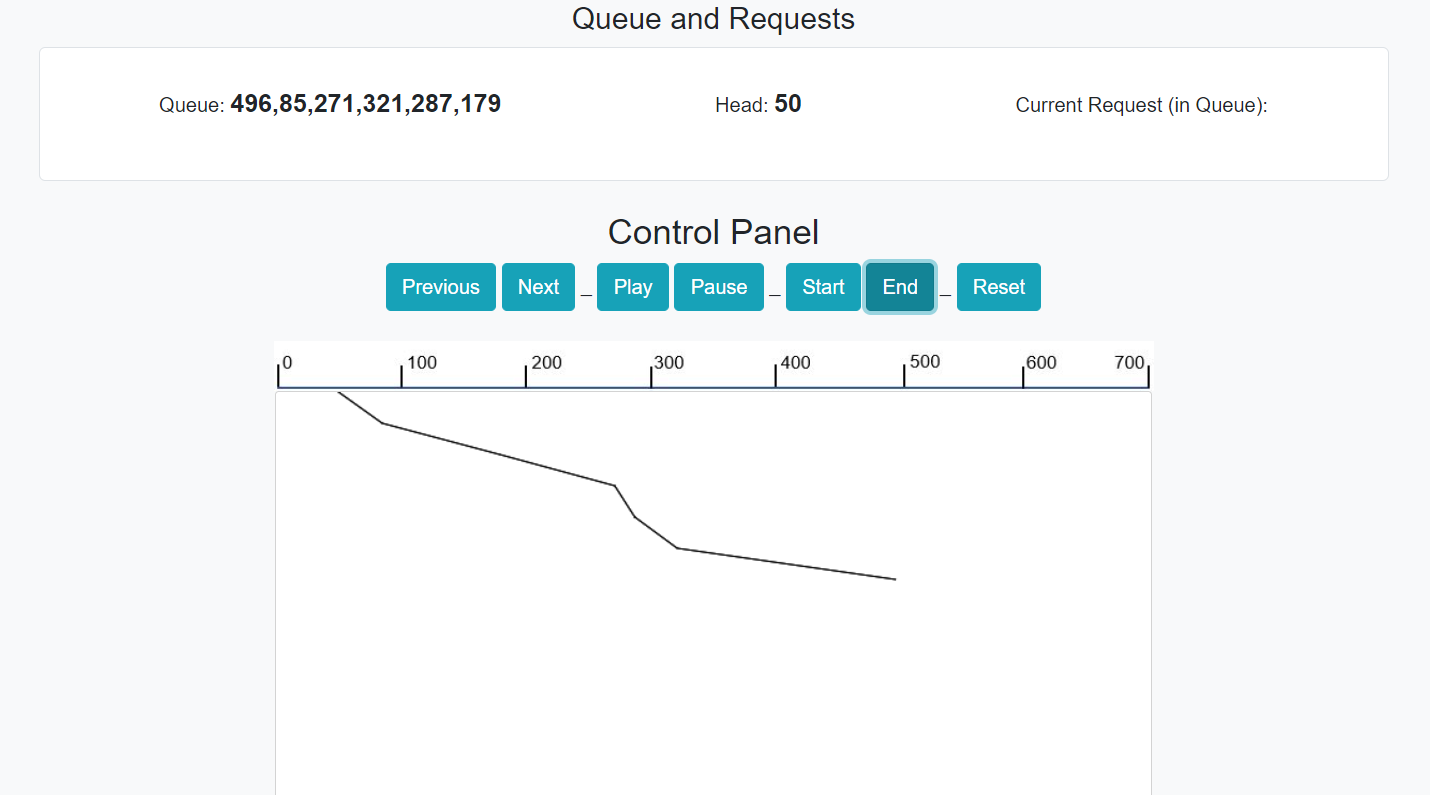
* Upon Control Panel, next button clicked until second to last Request was shown, displaying the proper location in queue for CSCAN, in this example this is 6, according to output.txt.
* The graph displayed the circular feature of CSCAN, past 633 in queue, the graph moved to one side of the disk to the other, as expected.
* All other buttons on Control Panel worked as expected for CSCAN algorithm.

**5.4 Test Function 4: LOOK**

Disk Scheduling LOOK Algorithm:



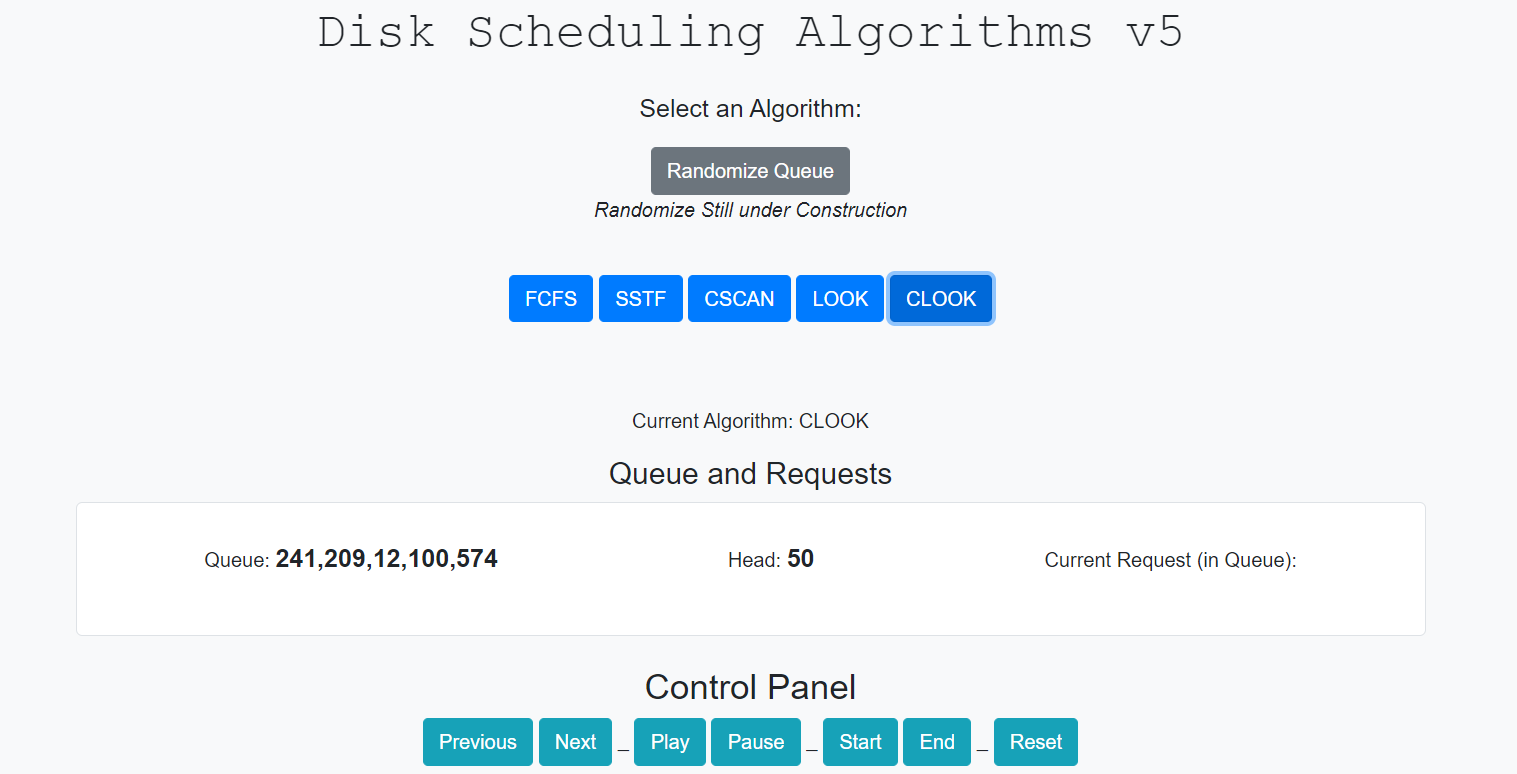
* Upon button click, LOOK input.txt and output.txt are retrieved.
* Queue and Head from input.txt are displayed, Current Request is awaiting user input to control panel.



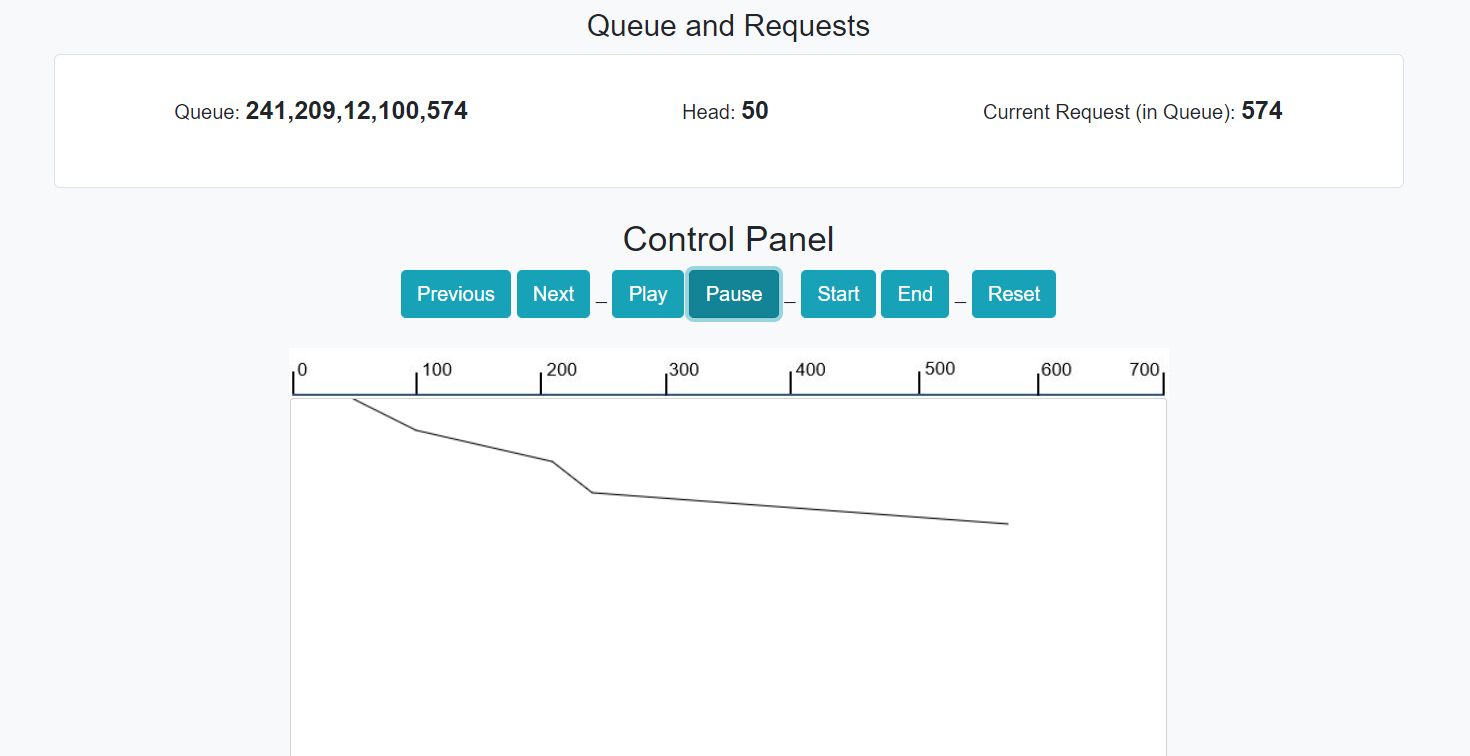
* Upon Control Panel, end button clicked to display graph at the end of queue, displaying the proper location in queue / graph for LOOK, according to output.txt.
* All other buttons on Control Panel worked as expected for LOOK algorithm.

**5.5 Test Function 5: CLOOK**

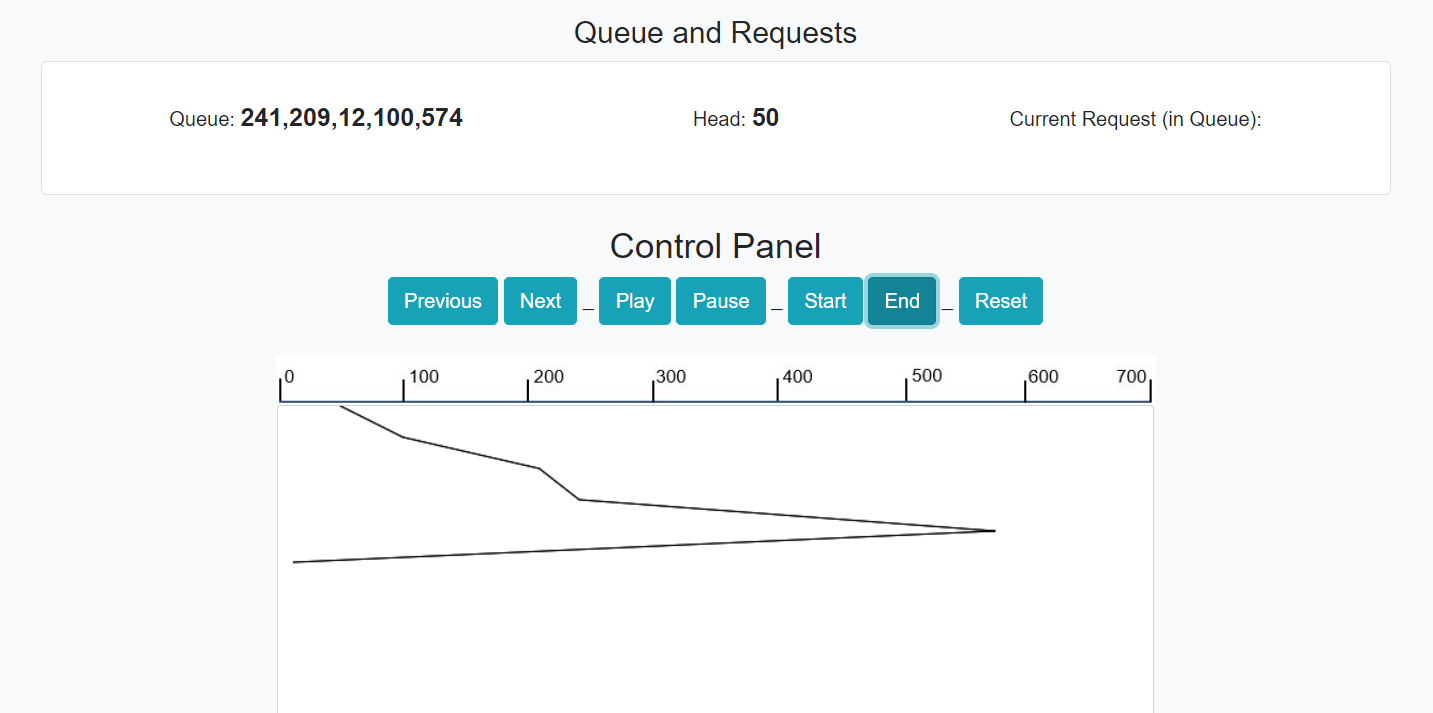
Disk Scheduling CLOOK Algorithm:



* Upon button click, CLOOK input.txt and output.txt are retrieved.
* Queue and Head from input.txt are displayed, Current Request is awaiting user input to control panel.



* Upon Control Panel, play and pause button were used to display the queue right before circular feature of CLOOK occurred. Play / Pause displayed the proper location in queue / graph for CLOOK, according to output.txt.

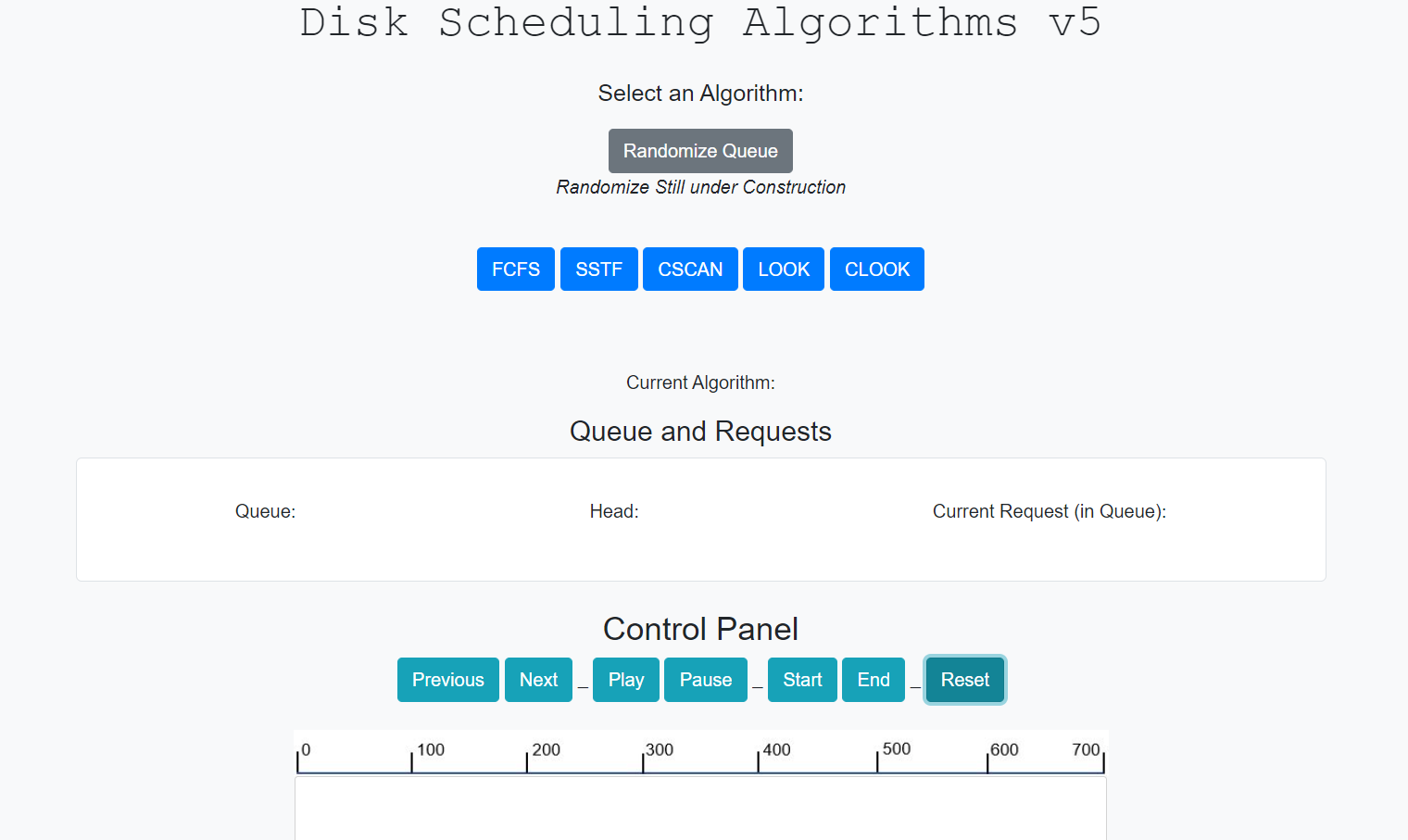


* Upon Control Panel, end button was used to display the queue right after circular feature of CLOOK occurred. End displayed the proper location in queue / graph for CLOOK, according to output.txt.
* All other buttons on Control Panel worked as expected for CLOOK algorithm.

**5.6 Test Function 6: Control Panel - Reset**

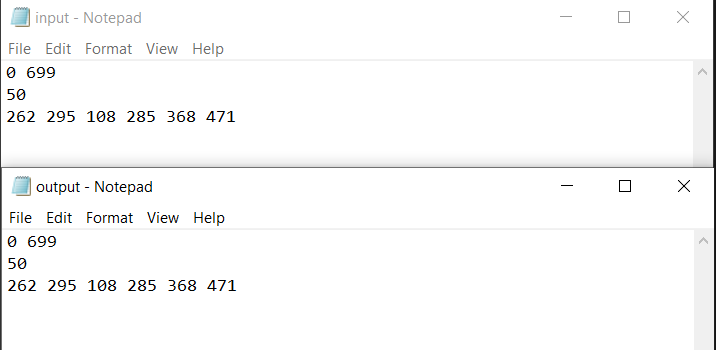
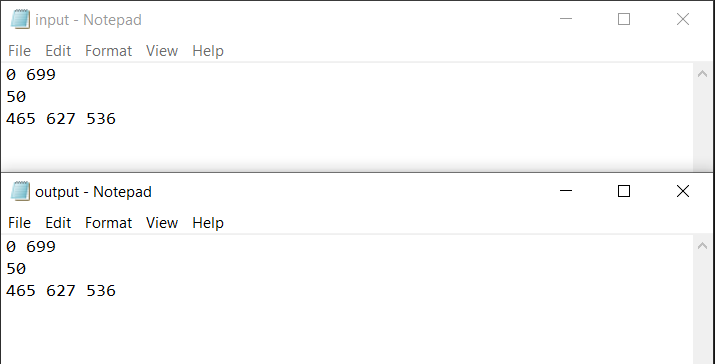
Reset used on Control Panel to Reset algorithm and algorithm choice:

* As seen with CLOOK the choice of algorithm, Queue and Request data, and Control Panel have been completely reset, as intended by reset implementation.

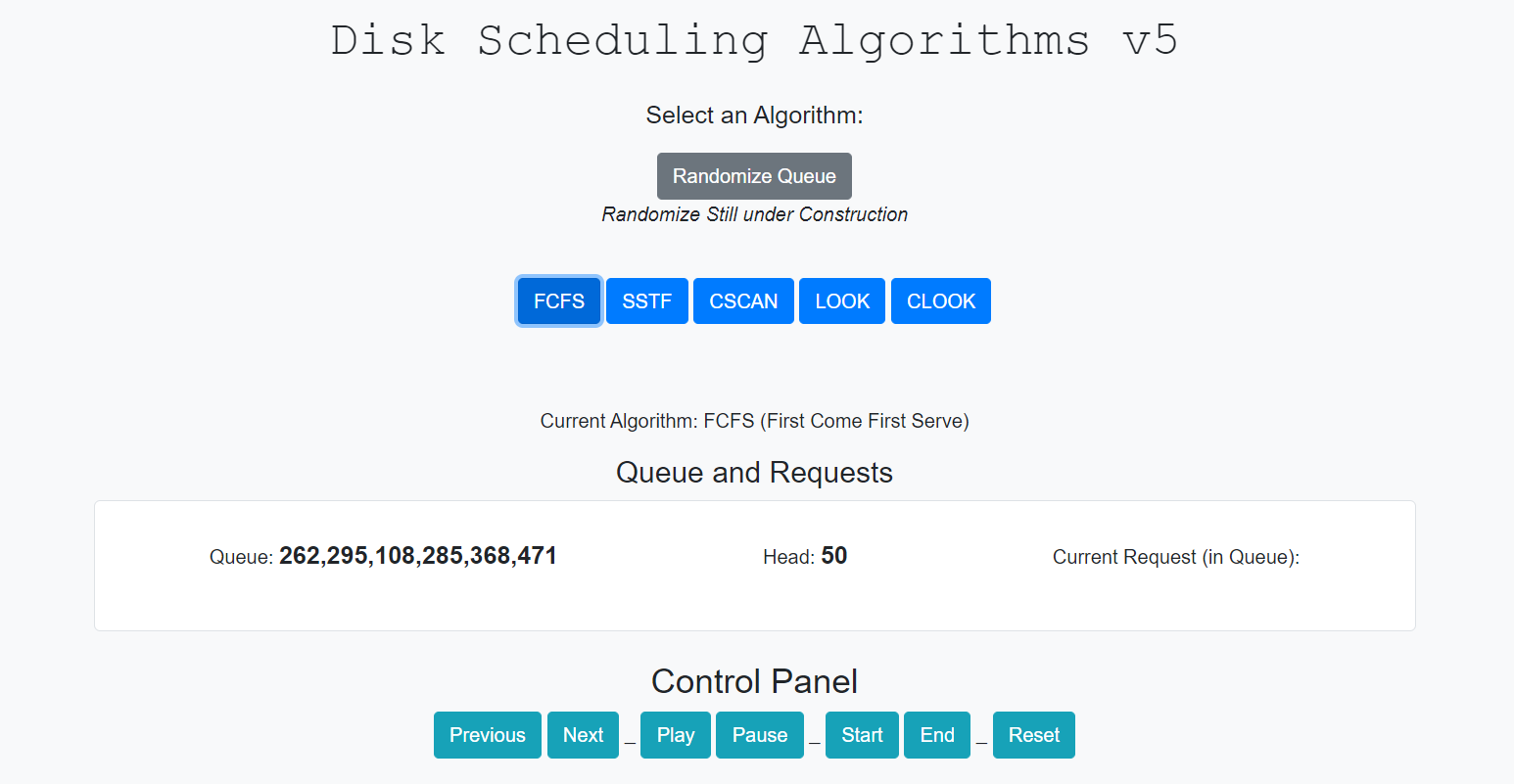


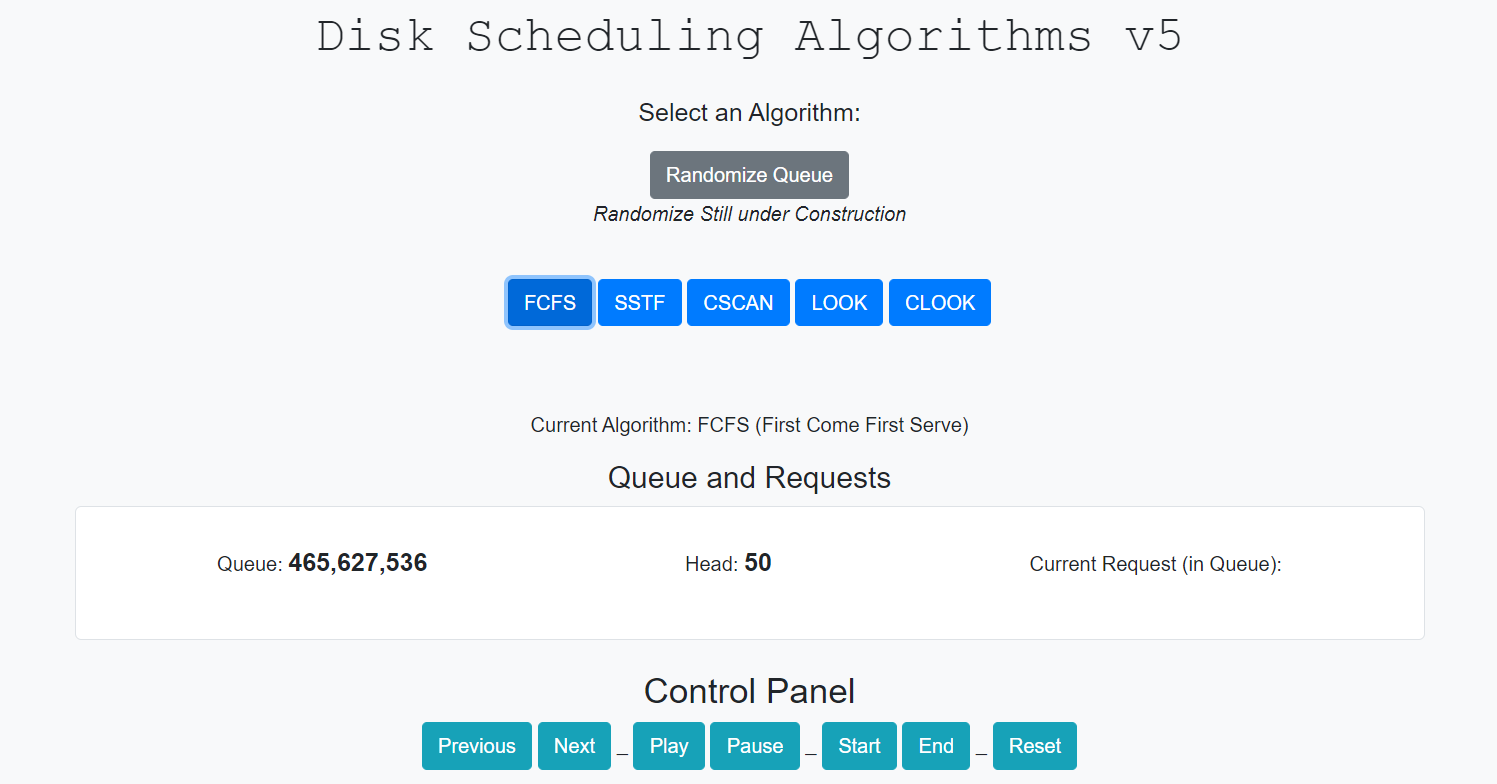
**5.7 Test Function 7: Randomize Queue**

Randomize used to change the queue values we are working with for each algorithm. This is through JavaScript to php to Java code back to php to JavaScript.

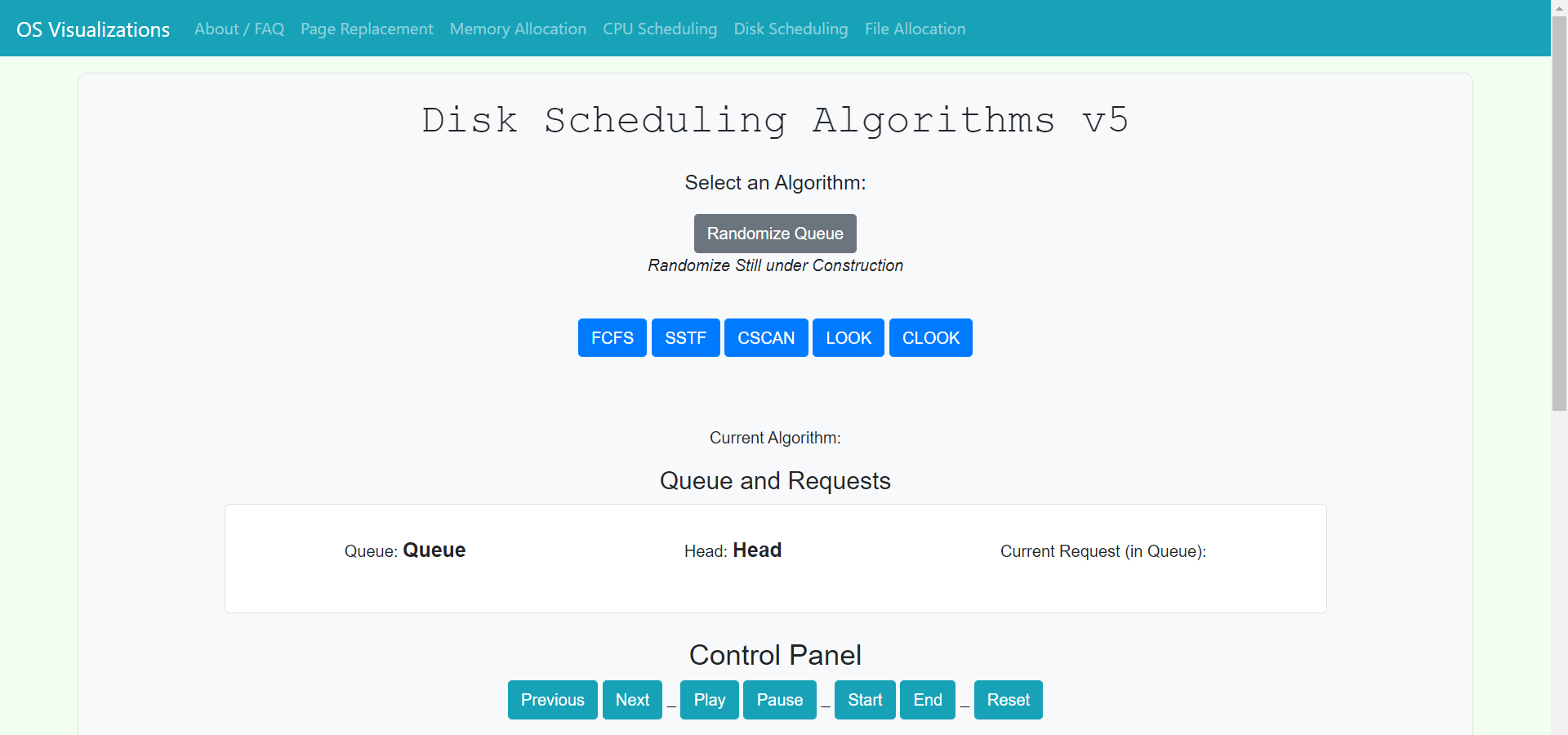
 

* As seen between before and after shots of input.txt and output.txt for FCFS, the values initially given are changed to new values and run through Java algorithms accordingly.

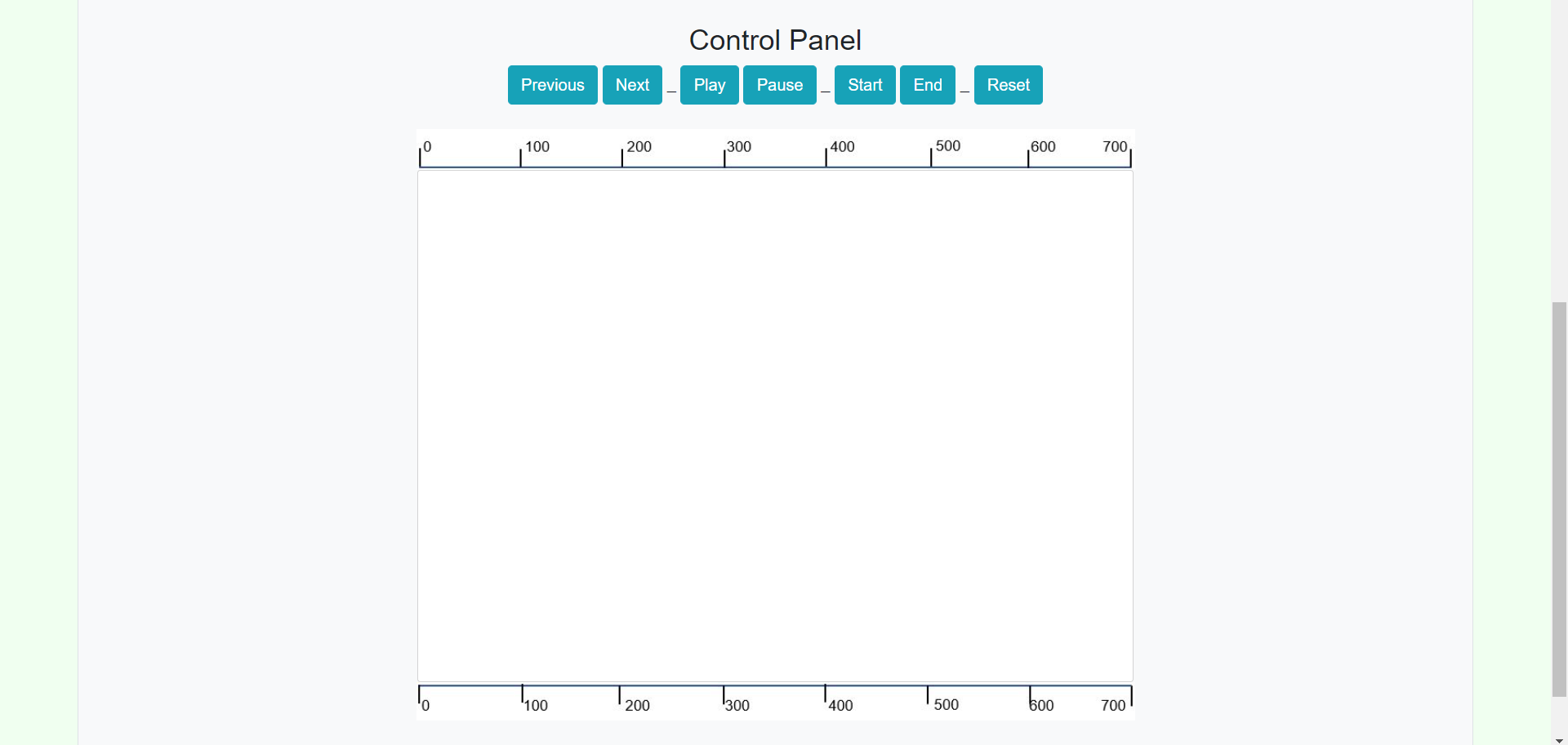




***\*\*Basic Layout of website for reference / control purposes:***



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\/ Continued \/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**6. Professional and Career Benefits**

**6.1 Challenges and Solutions**

There were various challenges that were presented to my group and myself, while working on this project. Some of these challenges and solutions to them are listed below.

**Challenge: Effecting HTML Canvas with Control Panel**

Solution:

Throughout my years as a computer science student, I had worked minimally with HTML Canvas. I first had to do research on HTML Canvas and how to connect it to JavaScript code that was being written to effect it. Once the research was complete, I went through a process of implementing and debugging each Control Panel Button that would directly affect the HTML Canvas (Graph) for Disk Scheduling. Each button took different coding techniques to implement and debug although I did end up managing to have a working on control panel to the standards, I had envisioned in my design early in the semester.

**Challenge:** **Clearing the Canvas through *Previous* button (Control Panel)**

Solution:

Every time I wanted to clear just the previous line made by next, it seemed to miss or not clear the correct area of the HTML Canvas. I fixed this through a matter of remembering where the line was printed and clearing them based on values [index-1], values[index], and different values for the vertical index which was changed for every new line written to the Canvas. This worked fine so the button worked synchronously with other buttons.

**Challenge: Animation / Autonomous movement through *Play / Pause* buttons**

Solution:

The play / pause buttons were an integral role in the development in the Disk Scheduling Visualization as it allowed the user to see the algorithm moving in real time to determine was it reads and writes to the disk. This was done by using a set interval function under an action listener. The action listener listening to when a Play / Pause button was clicked. This would start the interval counter and based on this count, it would print in real time through the next function.

**Challenge: Writing the Algorithms**

Solution:

Writing the algorithms was difficult, mainly due to my writing them in JavaScript the first time around. I had to research JavaScript as this was the first time, I had used this much of it in one project and gone this in depth with the code. Once, I had researched the algorithms and JavaScript I began writing them in JavaScript, unknown to me that the instructor wanted them specifically on the back end, which meant I would need to rewrite these algorithms in Java or PHP. I chose Java as this would take less research as I am comfortable in Java programming. Once rewritten the algorithms ran successfully on the backend and the next challenge regarding them was to connect them in some way to the front end.

**Challenge: Connecting the Back End Algorithms to the Front End**

Solution:

This was a difficult task for me to accomplish as I had barely worked with PHP and AJAX before this semester and project. I used AJAX placed within my JavaScript and PHP code placed in the folder where my specific algorithm was located to call my algorithms code. This PHP code would also call the randomize java code in the later challenges. By using a combination of AJAX and PHP I could successfully run the Java code to read, sort, and write the codes from input.txt to output.txt correctly by a click of a button on the front end, by the user.

**Challenge: Randomizing the inputs for each Algorithm**

Solution:

I used java code to build a randomizer for the queue data in input.txt for each algorithm’s datasets. This code would be accessed similarly to what was mentioned in the previous challenge, through the AJAX / PHP method. This method worked fine for calling and rewriting the input.txt and output.txt files, although there would still need to be a refresh on the front-end side of these codes, in order for them to be displayed properly on the website. On the server side this worked as intended, but there could still be worked done for the user to have less trouble on the front end displaying the new randomized data sets. Currently it works although there is the refresh work around that could be optimized for the user.

**Challenge: Reading input and output files on the front end**

Solution:

I have never worked with File I/O inside JavaScript code prior to this project although the result of reading back-end files on the front end worked as intended. I used JavaScript code with a declaration of an XML Http Request to read the files on the front end. These files were read into arrays to store the different variables of both the head and queue for input and output files. The website was updated to display the data read in by these functions.

**Challenge: Web Design of Website, Appealing to User, User Interface**

Solution:

Based on previous knowledge of web design and what the main web framework and other groupmates were creating, my web design was created. This was done through the help of Bootstrap5 and CSS3 to create a dynamic and appealing web design for the Disk Scheduling page. This design worked harmoniously with other groupmates as well as the main page of the web framework.

**6.2 Other Learned Lessons**

Many Lessons were learned through the process of working on this project. These lessons had coding and learning benefits as well as career and professional benefits. I learned a better process of debugging code through console logging variables on the website and inspecting the console to determine what went wrong. This technique allowed me to find problems and fix them quicker and identify more when they occurred. This has practical benefits in this project as well as in a professional setting where problems in code and other activities happen all the time, it helps to have the console ready to display what the computer sees as a second pair of eyes. I learned the importance of communication in a project setting, as asking a lot of questions and getting feedback leads to a better product for the client. It may seem daunting or annoying at times to ask a lot of questions, but it is necessary to understand and describe what exactly the client wants and what is even possible to develop in their image. Communication is key among group members as well. This furthers development along and fixes problems single developers may not have been able to fix on their own. It also seems to hold people to standards to hit project deadlines. I learned that writing out diagrams is important in communication between the developer and the client as well. These diagrams help as a framework for the developer to build something and the client to describe something. Diagrams and images seem to be more useful than words in describing a product.

Overall, I learned a lot through working on this project in both coding and project development tasks and challenges. These tips will help improve my work in the professional world via team building / developing and client communication in the future.

**7. Conclusion**

In conclusion, the OS Visualizations projects and my assigned portions came out as a success in fulfilling the instructions given by our supervisor at the start of this group project. In the business context, we created a platform that could be used as learning method for student and teachers alike to practice and review different processes surrounding practices in Computer Operating Systems. Many different algorithms were displayed, and different paths revealed across the knowledge base that we know of in Operating Systems today. We created proper user interfaces by website that emphasizes user connectivity, user experience, and user learning outcomes. We did research into new fields and found new discoveries to develop not only our software but our own knowledge of it. For my portions I progressed through meetings to talk about assignments and deadlines, to achieve the product pertaining to this report. I started out doing research and making designs for the website and its algorithms. Then, I moved into implementing those designs all the while continuing a line of communication to shape my implementations based on group and client demands. I tested and debugged software that was implemented in tandem to added new and rougher software to further myself to the final product. Working on fixing older implementations all the while designing and implementing new portions to meet client expectations. I then finished my testing and used web design techniques to chisel out a webs design friendly for the audiences mentioned in our business contexts. This was then connected to the back-end files and further tested to make sure the connection was strong between front and back on both local and live servers. Different web designs, different buttons, different versions of algorithms were all optimized over the weeks to produce what we currently can display. Multiple versions of our code were documented both on the live server and through weekly updates to display what we were working on and when in order to display progress. A lot of time and effort went into identifying and solving software problems in order make the software better and easier on the users intended to learn off our product. Based on all the time, effort, and group / client communication, I believe that we successfully delivered a working OS Visualizations Website with a well-documented web server that is presentable and fulfills the intensions our supervisor / client requested.

**8. References**

**General Research:**

Geeks for Geeks (Disk Scheduling):

<https://www.geeksforgeeks.org/disk-scheduling-algorithms/>

<https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/>

<https://www.geeksforgeeks.org/file-allocation-methods/>

<https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>

<https://www.geeksforgeeks.org/first-fit-allocation-in-operating-systems/>

*Various Operating System PowerPoint Slides from SUNY New Paltz CPS340: Operating Systems Lecture*

**Coding Components:**

Bootstrap5 (CSS Framework):

<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

JavaScript at W3 Schools (JavaScript Research):

<https://www.w3schools.com/js/DEFAULT.asp>

Play / Pause Buttons (Real-time setInterval() Research):

<https://www.youtube.com/watch?v=ubLC1JxMqfY&ab_channel=dcode>

HTML Canvas at W3 Schools (Canvas Research):

<https://www.w3schools.com/html/html5_canvas.asp>

Stack Overflow (Various Debugging Fixes and Components):

<https://stackoverflow.com/questions/585234/how-to-read-and-write-into-file-using-javascript>

<https://stackoverflow.com/questions/2128619/run-java-class-file-from-php-script-on-a-website>

Php Connections (Research and Problem Solving):

<https://www.w3schools.com/php/php_file_open.asp>

Aram Agajanian (Php connections through Wyvern Server):

**Acad. Dept. Support**  
Department: Academic Computing  
Office: REH 007  
Phone: (845) 257-3775  
E-mail: [agajania@newpaltz.edu](mailto:agajania@newpaltz.edu)

**Server Hosting / Testing:**

XAMPP (Local Server Hosting):

<https://www.apachefriends.org/index.html>

**9. Appendix**

Appendices reference further development information. This is if further development where to take place on the OS Visualizations Web page / Web Server.

**Appendix A: Further Development to Disk Scheduling Software**

This section is dedicated to those who may pick up this project in the future and would like a more detailed account of what specific portions of the code. Appendix A is an addition to the Developers Guide for specific coding components.

As the code is laid out at the final presentation of the Spring 2021, this is a detailed account of software following the information in section 4.1 and 4.3 of this report.

For Disk Scheduling index.html:

* **Inside <head> Tag**:
  + Proper meta and script tags to include HTML5, CSS, and Bootstrap5 functions
  + <style> Tag: Grid Container for Queue and Requests, Html Canvas and Img for Disk Graph, Various Text styling, and container length.
* **Inside <body> Tag:**
  + Declaration of Navbar element and links (Bootstrap Styling)
  + Header with Button for data randomization (Bootstrap)
  + Select and Algorithm with Algorithm Buttons (Bootstrap)
  + Grid container declaration for Queue and Requests Text
  + Control Panel with Control Panel Button Elements (Bootstrap)
  + Img elements which place graph bars, Html Canvas Declaration in between Img elements
* **Inside <script> Tag:**
  + *Note: This is still inside body tag.*
  + Various global variable declaration
  + setInterval() Button Listener for Play / Pause Feature
  + Reset Button Function: Resets Relevant data to current algorithm
  + FCFS, SSTF, CSCAN, LOOK, CLOOK, Button call to various connections to functions
  + Randomize inputs: Done through .ajax call appropriate .php for algorithms
  + Text Reader Functions: These functions read each algorithms input and output text files and returns the data for webpage outputs.
  + Next, Previous, Start, and End Button Functions: These functions create action for the corresponding buttons through moveto(), lineto(), stroke(), and clearRect() calls on the canvas .

**Appendix B: Further Development: Problem Discovery**

Various Challenges / Problems were listed in section 6.1 of this report, although there are some problems / challenges that went unoptimized and not fully research, in the essence of the time constraints. Some of these Problems that may be solved by further development include:

* Better data manipulation on the backend of the Wyvern Server
* Working out permission problems between web page and web server on the Wyvern Server
* More in dept php connections between portions of the website / Disk Scheduling.
* Better live animations on the HTML Canvas for Disk Graph.
* Various other algorithms for Disk Scheduling as well as other groupmate portions.
* Better utilization of animations through jQuery.