OPERATING SYSTEM

TEAM 14: DISK SCHEDULLING ALGORITHMS & PAGE REPLACEMENT AND BELADY'S ALGORITHMS

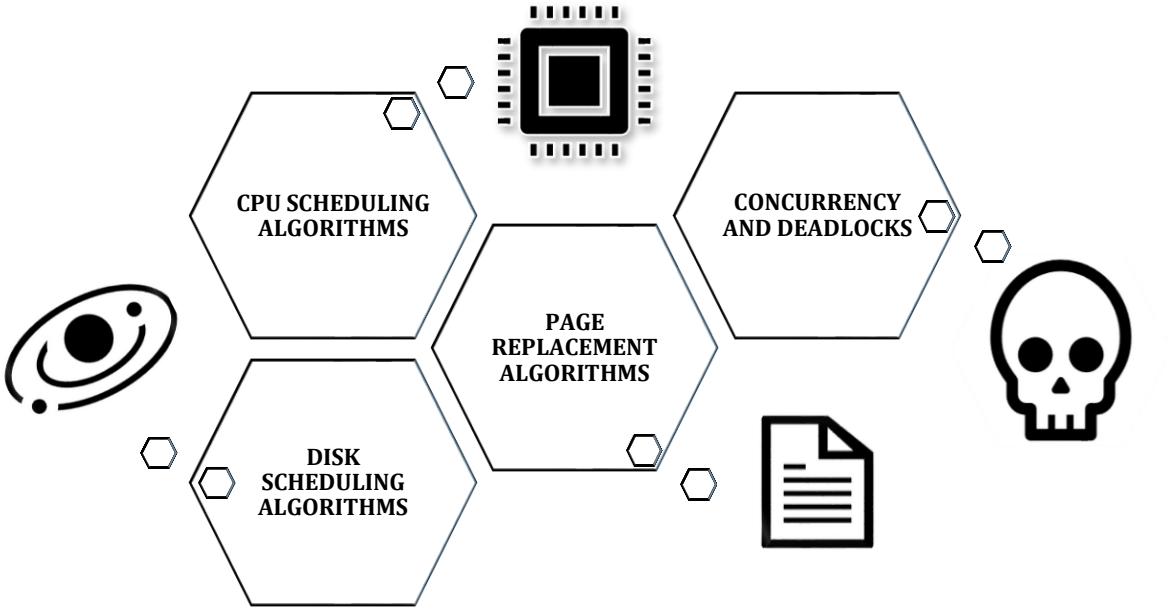
- **BADAL PARMAR 18BCP011**
- KHUSHI KATARIYA 19BCP068
- KALGI SHAH 19BCP062
- HETANSHI DABHI 19BCP053
- ADITYA GANDHI 19BCP002

Table of Contents

ABOUT PROJECT

OPERATING SYSTEM:

An **Operating System (OS)** is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.



Our Project is based on the implementation of the two algorithms which are: Disk Scheduling Algorithm and Page Replacement Algorithm which was implemented by our seniors.

We made a GUI for the implementation of both the algorithms by using JAVA programming language. This was done in continuation to the project done by our senior on Page Replacement Algorithm by using JAVA programming language on Eclipse IDE. We merged both the projects and made one single platform for all the 4 main algorithms of Operating System.

In this project, we are going to focus on the Page Replacement Algorithms & Disk Scheduling Algorithms. We are going to show how the algorithms are implemented, we are going to provide you with algorithm calculator, and a comparison graph simulator, that helps you to compare between the all the algorithm for the similar kind of input.

Disk Scheduling Algorithms

Disk Scheduling is basically used to schedule I/O requests arriving in the system. It is important to schedule the I/O requests because at a time many I/O requests are made, and our disk controller can execute any one of the requests. So, to choose one request out of the others we use various disk scheduling algorithms so that each request is accessed in the minimal time and each request gets executed.

There are many Disk Scheduling Algorithms but before discussing them let us have a quick look at some of the important terms:

SEEK TIME

Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written. So, the disk scheduling algorithm that gives minimum average seek time is better.

ROTATIONAL LATENCY

Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So, the disk scheduling algorithm that gives minimum rotational latency is better

TRANSFER TIME

Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

DISK ACCESS TIME

Disk Access Time = seek time + rotational latency + transfer time

DISK RESPONSE TIME

Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of all the requests.

6 Algorithms Of **Disk Scheduling Algorithms** are explained and implemented in this project which are as follows:

DISK SCHEDULING ALGORITHMS

- **FIRST COME FIRST SERVE (FCFS)**
- **SHORT SEEK FIRST TIME (SSTF)**
- **SCAN**
- **C-SCAN**
- **LOOK**
- **C-LOOK**

→ **FCFS (First Come First Serve)**:

↳ In FCFS (First Come First Serve), the requests are processed in the order in which they arrive. In this algorithm, starvation does not occur because FCFS addresses each request.

- In FCFS disk scheduling, there is no indefinite delay.
- There is no starvation in FCFS disk scheduling as each request gets a fair chance.
- FCFS is inefficient as compared to other algorithms.
- In FCFS, scheduling disk time is not optimized.
- Results in increase of total seek time.

→ **SSTF (Shortest Seek Time First)**

↳ SSTF stands for Shortest Seek Time First. In SSTF, requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request that is near the disk arm will be executed first. SSTF is certainly more efficient over FCFS as it decreases the average response time and increases the throughput of the system. It breaks the tie in the direction of head movement.

→ **SCAN**

- In the SCAN disk scheduling algorithm, the head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reaching the other end. Then the direction of the head is reversed, and the process continues as the head continuously scans back and forth to access the disk. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

→ **C-SCAN (Circular Scan)**

- The Circular SCAN (C-SCAN) scheduling algorithm is a modified version of the SCAN, that deals with the inefficiency of the SCAN algorithm by servicing the requests more uniformly. Like SCAN, C-SCAN moves the head from one end servicing all the requests to the other end. However, as soon as the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip and starts servicing again once it reaches the beginning.

→ **LOOK**

- LOOK is the advanced version of SCAN (elevator) disk scheduling algorithm. The LOOK algorithm services request similarly as SCAN algorithm meanwhile it also “looks'' ahead as if there are more tracks that are needed to be serviced in the same direction. If there are no pending requests in the moving direction the head reverses the direction and starts servicing requests in the opposite direction.

→ **C-LOOK (Circular Look)**:

- C-LOOK is an enhanced version of both SCAN as well as LOOK disk scheduling algorithms.
- In this algorithm, the head services requests only in one direction (either left or right) until all the requests in this direction are not serviced and then jumps back to the farthest request on the other direction and service the remaining requests which gives a better uniform servicing as well as avoids wasting seek time for going till the end of the disk.

PAGE REPLACEMENTS ALGORITHM

Page Replacements algorithm is one of the most important concepts of the Operating System. The major drawback of any operating system is its speed and memory allocation process. The number of memory calls done to access frequently used pages, is tiresome and affects the speed, accuracy and durability of the computer hardware and the other software present in the system.

→ **3** Algorithms Of **Page Replacements Algorithm** are explained in this project which are as follows:

- **FIFO (First In First Out)** Page Replacement Algorithm
- **LRU (Least Recently Used)** Page Replacement Algorithm
- **OPR (Optimal Page Replacement)** Page Replacement Algorithm.

→ **FIFO (First in First Out) Page Replacement Algorithm**

First In First out Page Replacement Algorithm, is the simplest algorithm. Page Replacement Algorithm follows a simple queue of pages entering the page frame. The oldest page is at the head of the queues. Thus, when the need arises the page at the head of the queue is removed and replaced with another set of frames.

→ **LRU (Least Recently Used) Page Replacement Algorithm**

Least Recently Used page replacement algorithm is the second most optimal algorithm and can be implemented easily. According to the LRU algorithm, when the need arises to replace the page in the frame stack, the least recently referred page is replaced.

→ **OPR (Optimal Page Replacement) Algorithm**

Optimal Page Replacement is the most efficient algorithm but cannot be implemented practically in an operating system. As, in the operating system we do not have finite number of page references and hence not possible in practice as the operating system cannot know future requests.

GUI (GRAPHICAL USER INTERFACE):

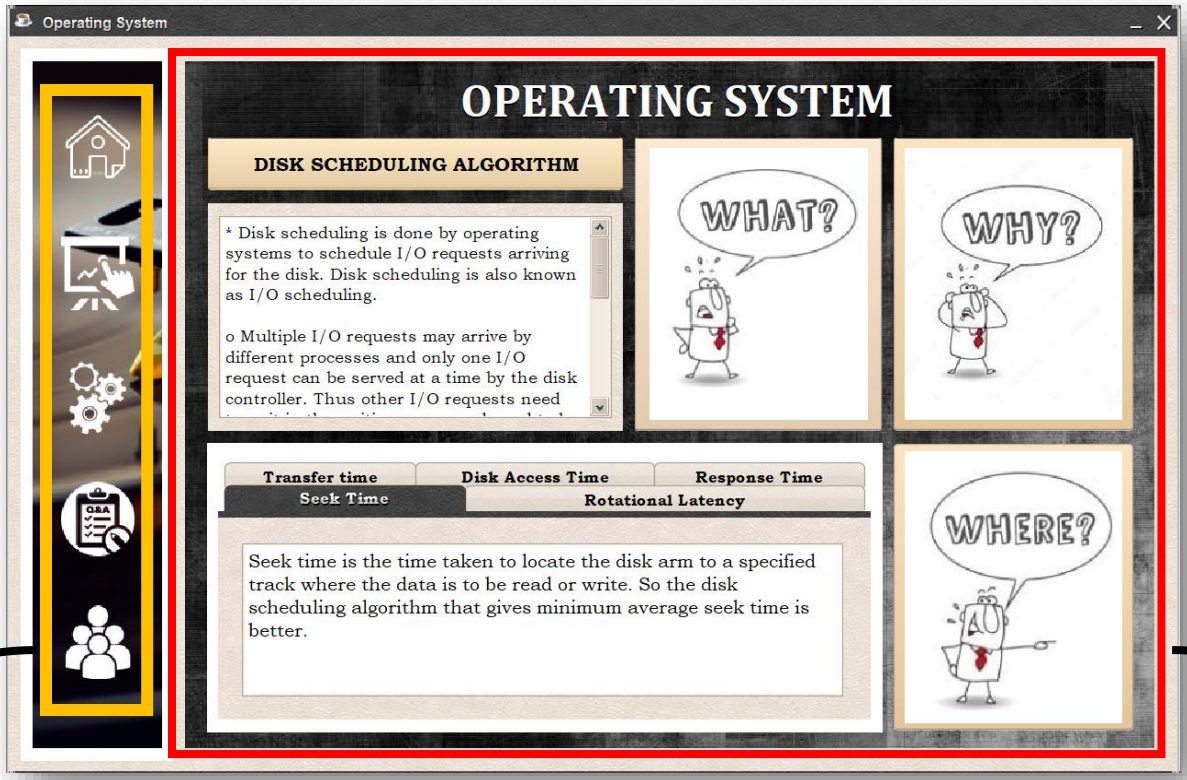
- To implement all the algorithms and explain the working of the algorithm visually our team has created a GUI. The GUI is a Desktop App developed using Java Programming Language. Whole code has been developed on the Eclipse IDE Java Editor. All the methods of downloading the IDE and external jar files, also setting up the IDE and jar files and running the code has been mentioned in the README file attached with this file. The README file will guide the student to set up the editor and external jar file and help them to run our code snippet effortlessly.
- Now, attaching the snapshots of GUI and briefly explaining all the components of our GUI.

LAUNCHING OF THE DESKTOP APP

- Execution of GUI will bring one to a splash screen as shown below. There is a **progress bar** at the bottom which when reaches **100%** opens the **Homepage** frame of the project.

SPLASH SCREEN

HOMEPAGE



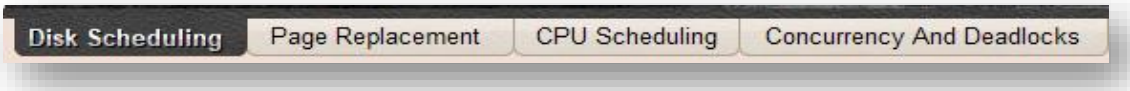
SIDE PANEL:

1. Homepage
2. Algorithms
3. Simulation
4. Quiz
5. About Us

MAIN PANEL:

1) What the Algorithms used for?
2) Why the Algorithms used for?
3) Where the Algorithms used?
4) Disk Scheduling Algorithm (Introduction)
5) Important terms for Algorithms

1) **What? Tab** explains about the 4 different modules of Operating system.
2) **Why? Tab** explains about the need of the algorithms are which kind of algorithms are implemented.
3) **Where? Tab** tells us the alternative applications of algorithms apart from the Operating System.



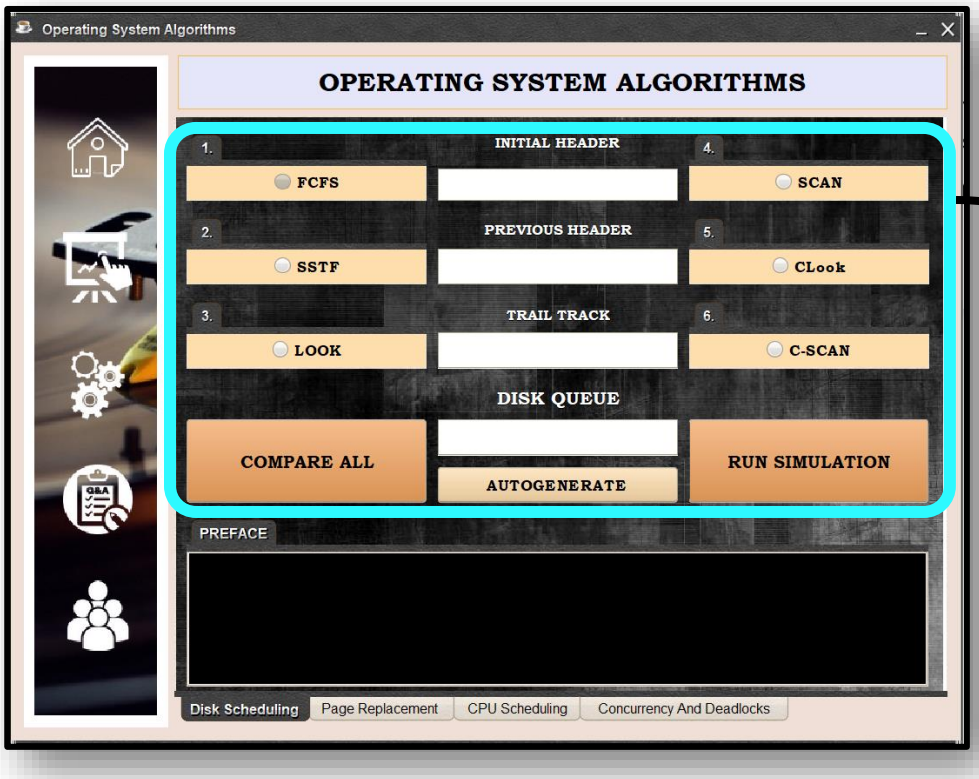
TABBED PANE

For implementation of disk scheduling algorithms in the GUI made by seniors we had to make a separate GUI first and integrate it at last with the seniors' GUI.

So, in our final GUI we made sure that it would be easier for our juniors to add more concepts to the GUI rather than building a separate one and integrating at last. For this, tabbed panes were implemented using which one can toggle between frames.

ALGORITHM TAB

The Algorithms tabs implement all the algorithms dynamically and provides the graphical output where we compare all the algorithms for the similar input. The Algorithms also contains an **autogenerate** button that helps to generate values automatically and thus helps the students to execute examples or sums multiple amounts of times without lag.



RADIO BUTTON SELECTION FOR ALGORTIHMS

In the algorithm panel, there are 6 radio buttons for all 6 algorithms of Disk Scheduling algorithms. You can choose any function and write down the values required in the text field and then click on **RUN SIMULATION** button to display its graph and seek time. You can also put values in disk queue automatically by clicking on **AUTOGENERATE** button. Below **PREFACE** panel is kept which displays brief information about algorithm selected.

Algorithm implementation of Disk Scheduling:

As shown in precious figure the main panel of Algorithm GUI is divided in 2 section: Algorithm Selection/Implementation and Preface

ALGORITHM SELECTION/IMPLEMENTATION SECTION:

- 6 Algorithms Radio Button Selection
- Initial Header, Previous Header and Trail Track
- Autogenerate Button for generating disk Queue
- Compare ALL and Run Simulation Button
- **The implementation of Algorithm will be resulting into Graph of their respective Seek Time.**
- **Compare All button will take user to Comparison frame where the comparison of all algorithms is done keeping any one as reference.**

PREFACE

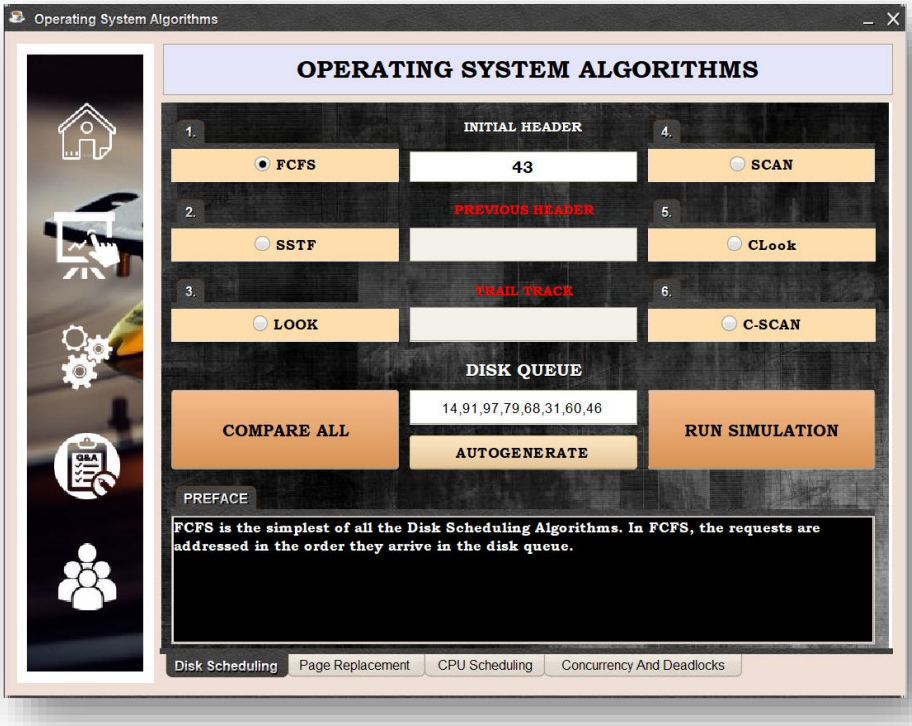
On selection of any radio button a small brief for respective algorithm will be shown

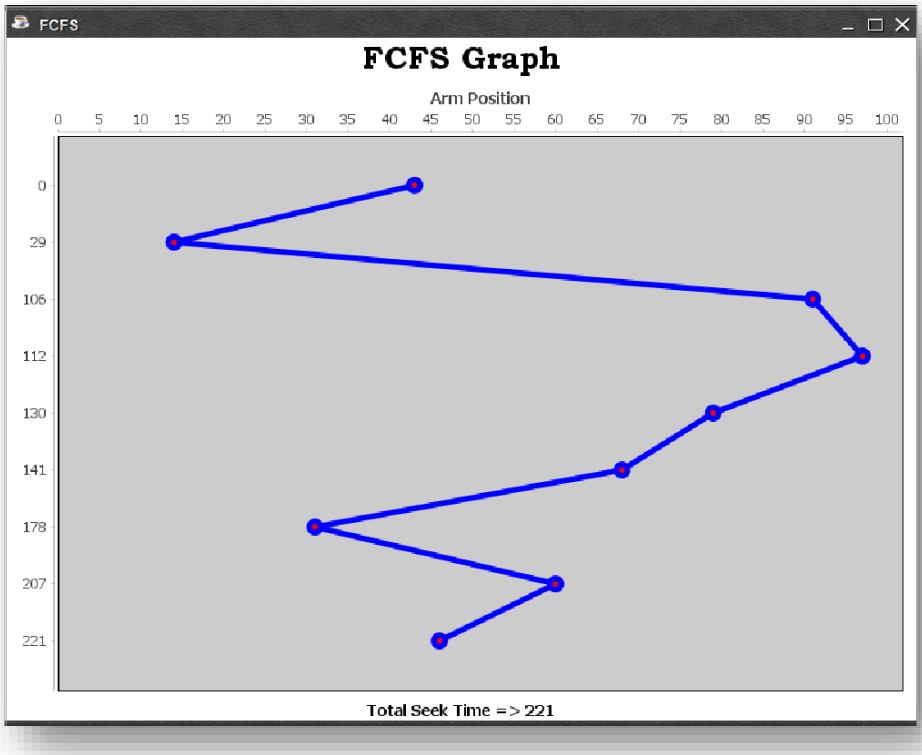
All the implementation of algorithms is shown as follows:

- **The values of Disk Queue are generated through AutoGen and algorithms are ready for implementation.**
- **Below given are the seek time graph for respective values:**

→ FCFS (First Come First Serve)

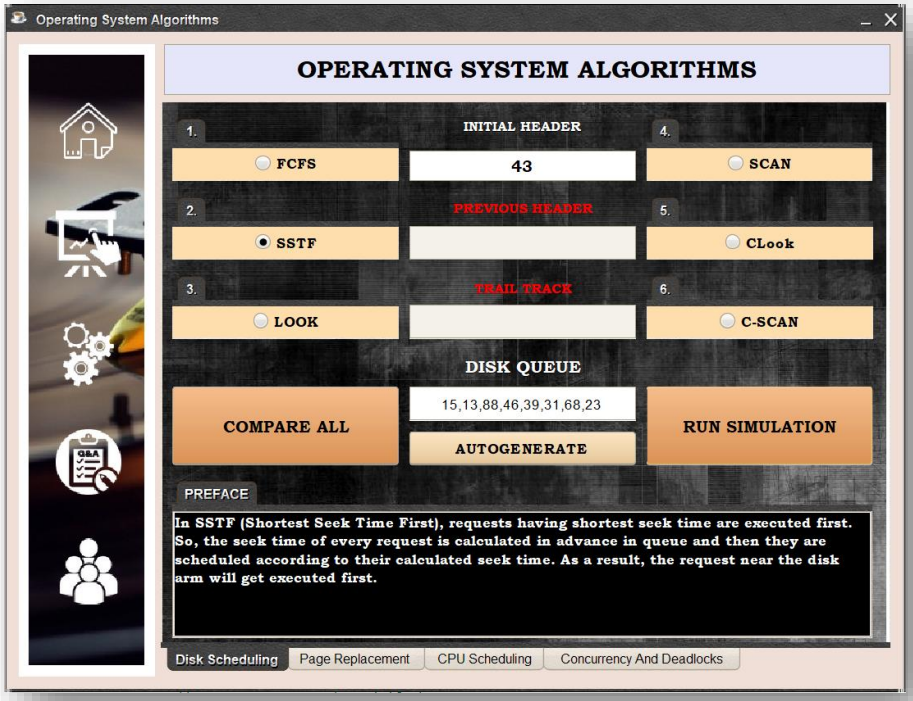
→ **FCFS is selected the preface shows its small description Previous header and Trail Track are not required therefore, highlighted with RED color.**

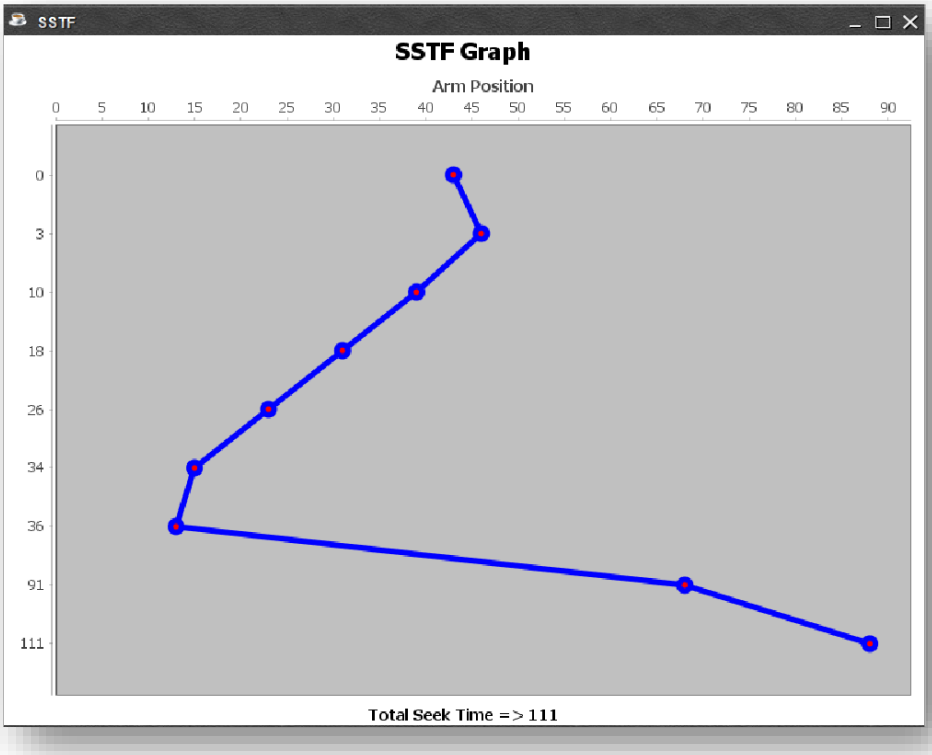




→ SSTF (Shortest Seek Time First)

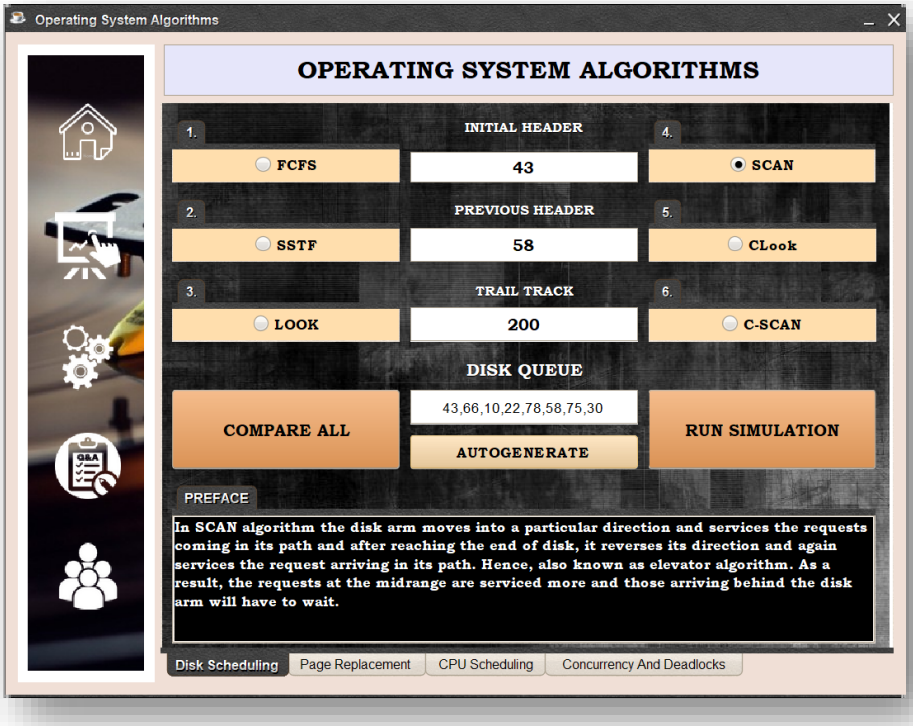
→ **SSTF is selected the preface shows its small description Previous header and Trail Track are not required therefore, highlighted with RED color.**

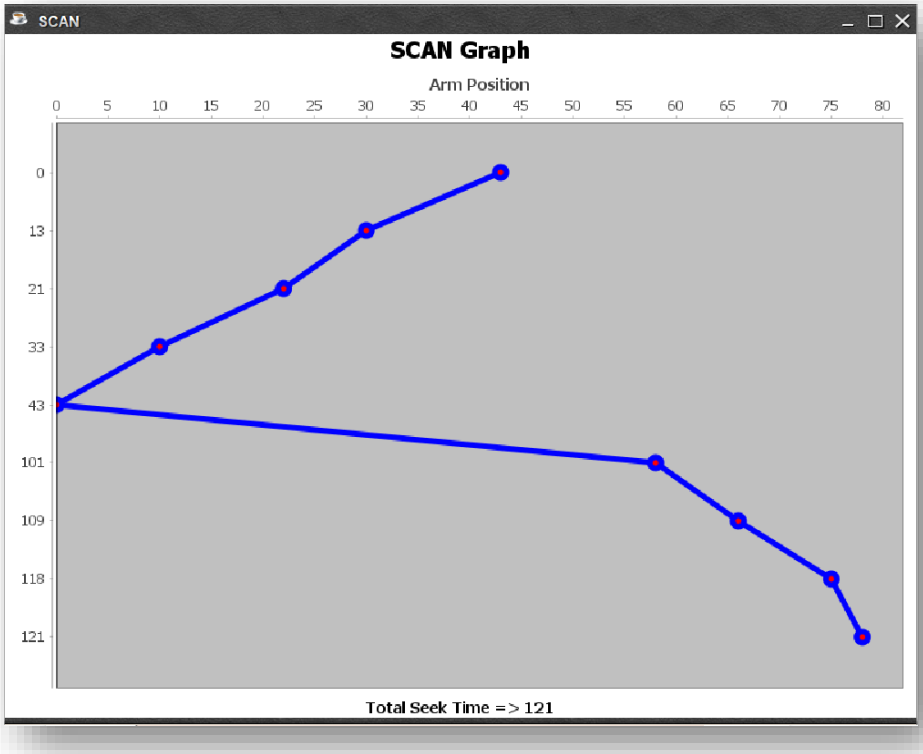




→ SCAN (Elevator Scan)

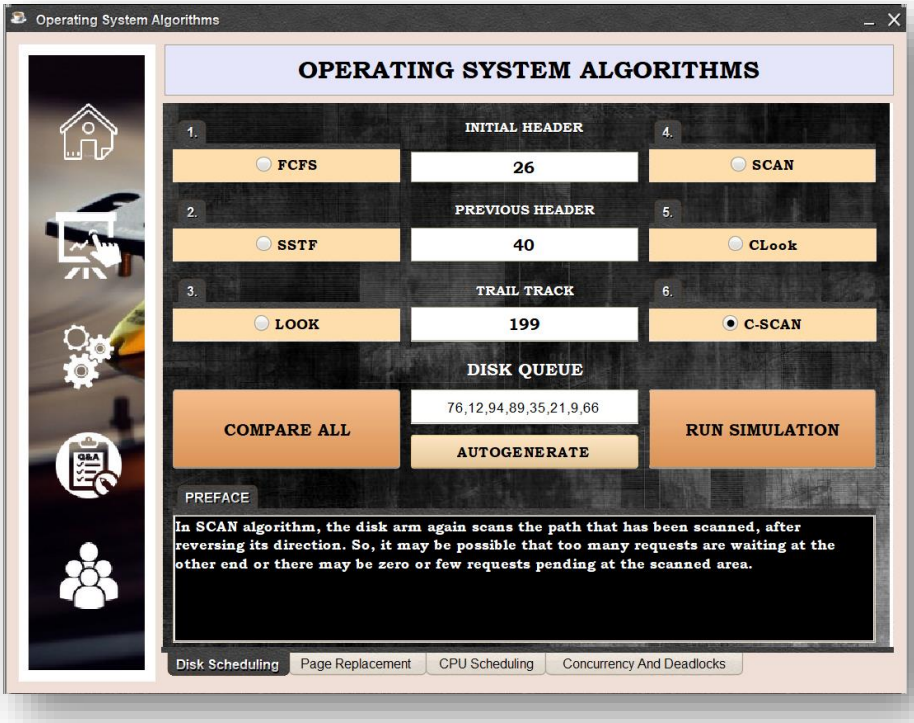
→ **SCAN is selected the preface shows its small description All the tabs are required Previous header is used to determine the direction if greater than initial header the Left to Right and if smaller than left to right.**



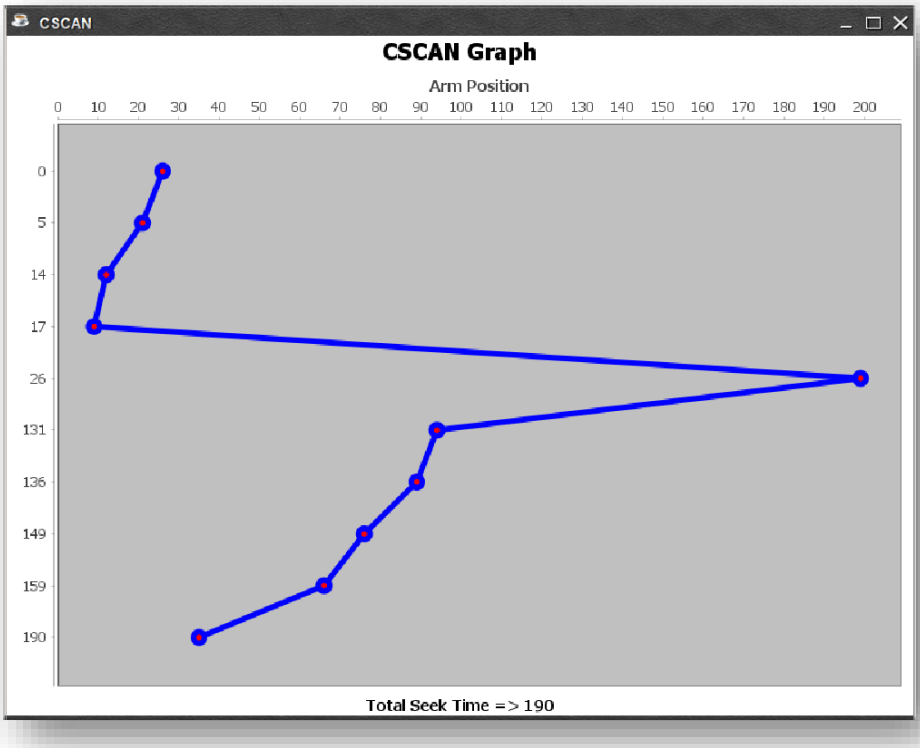


→ CSCAN (Circular Scan)

→ **CSCAN is selected the preface shows its small description All the tabs are required Previous header is used to determine the direction if greater than initial header the Left to Right and if smaller than left to right**

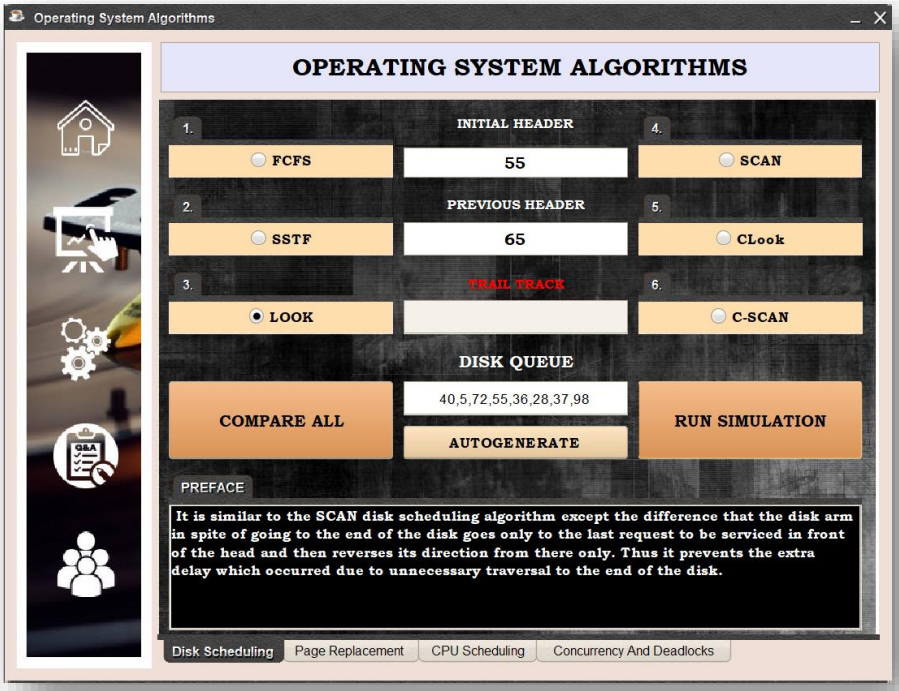


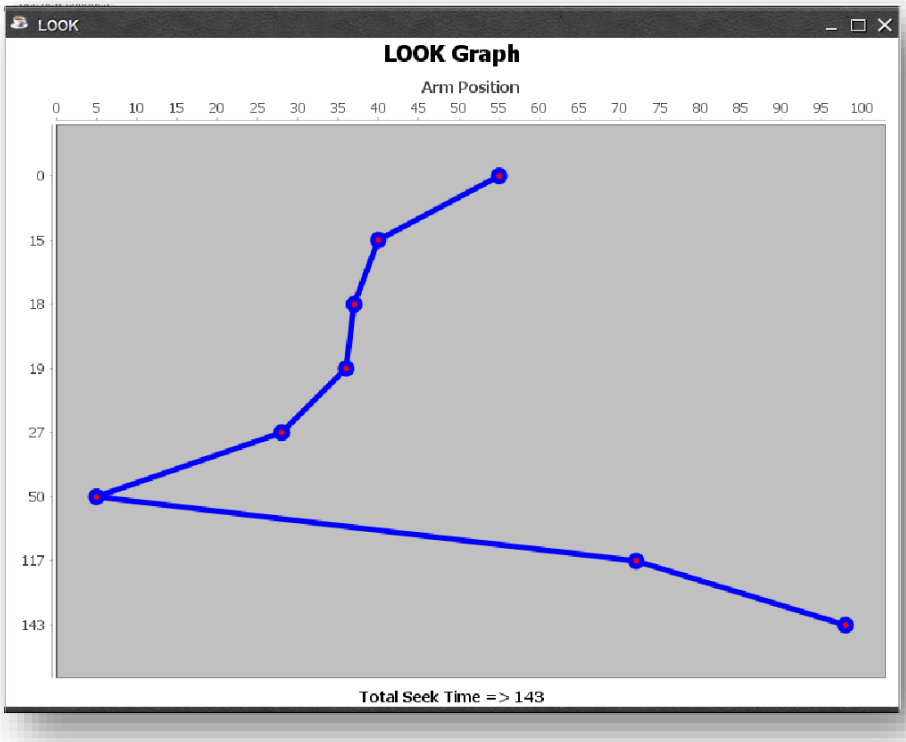
.



→ LOOK

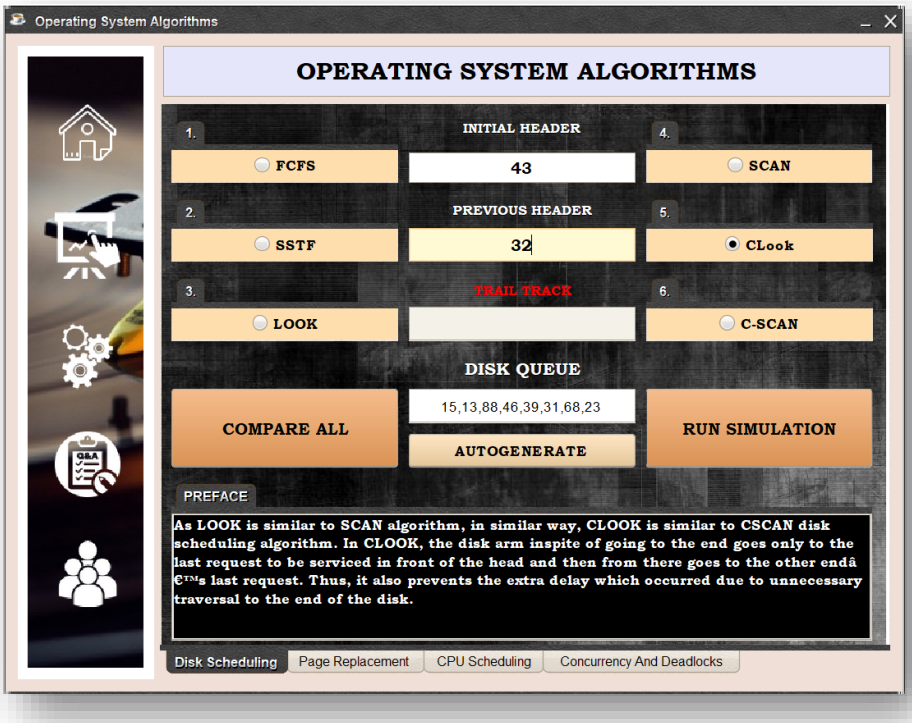
→ **LOOK is selected the preface shows its small description All the tabs are required Previous header is used to determine the direction if greater than initial header the Left to Right and if smaller than left to right**

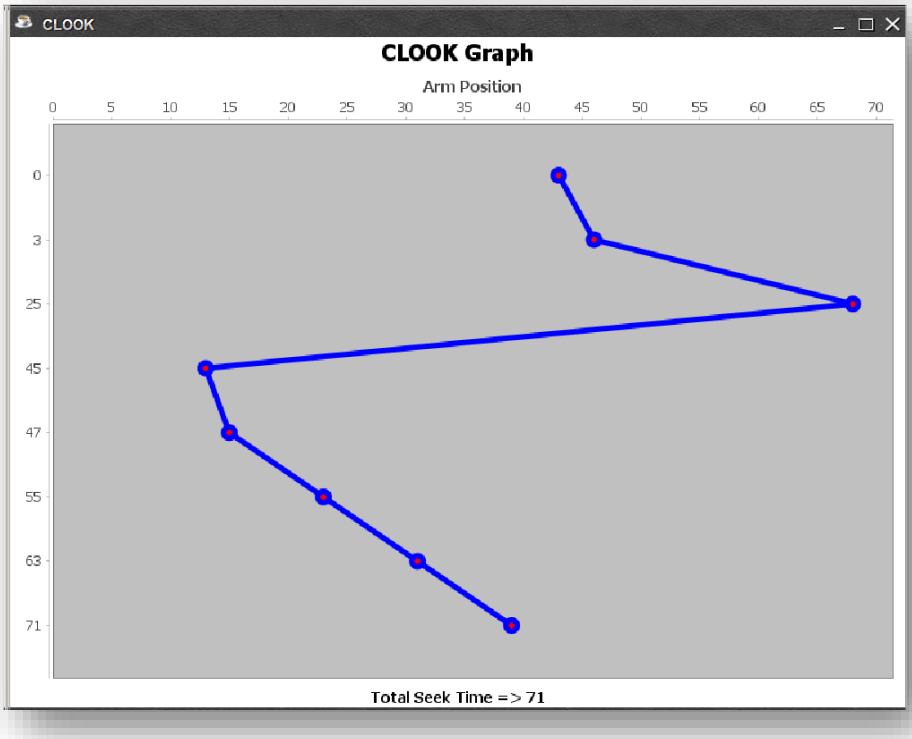




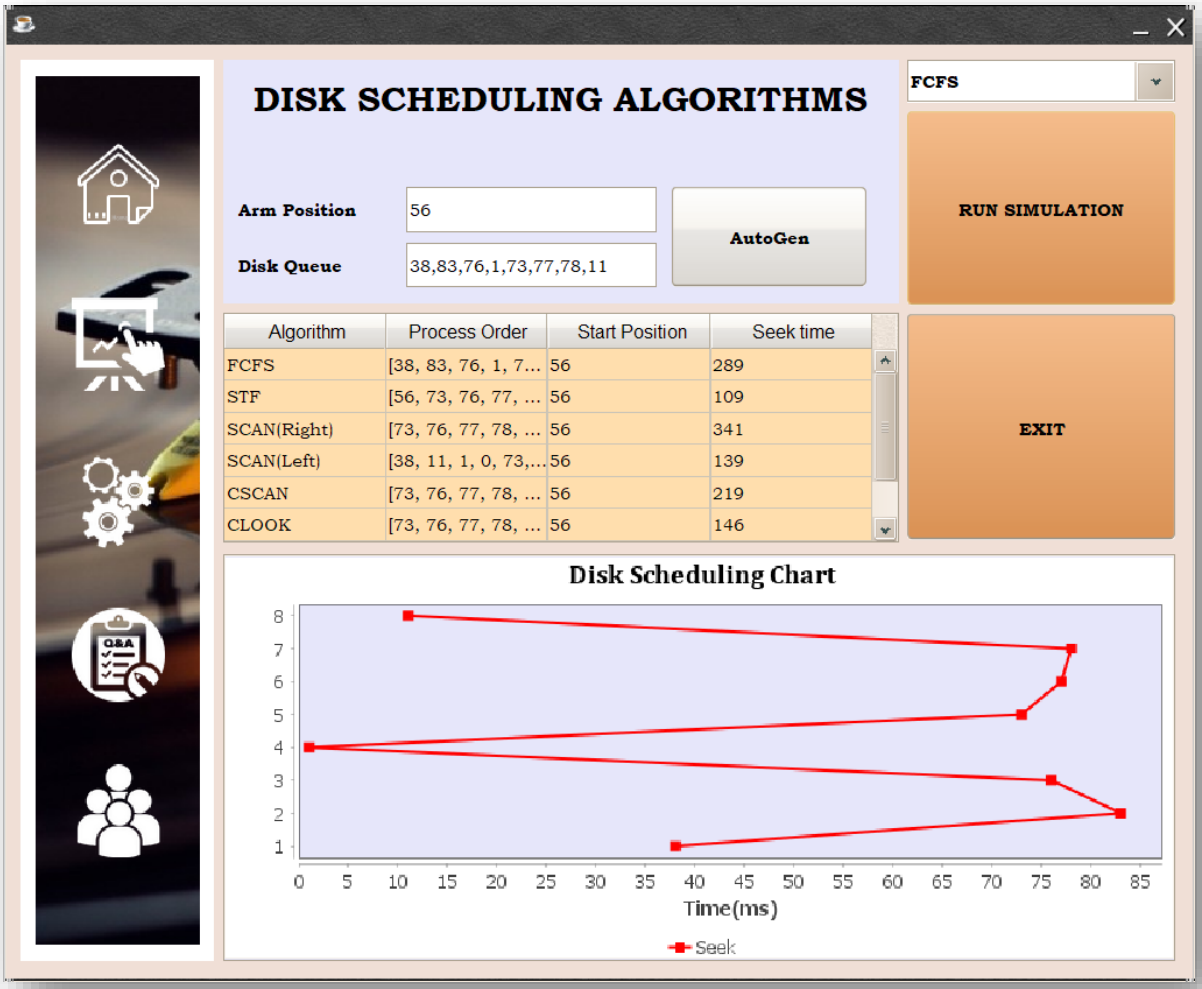
→ CLOOK (Circular Look)

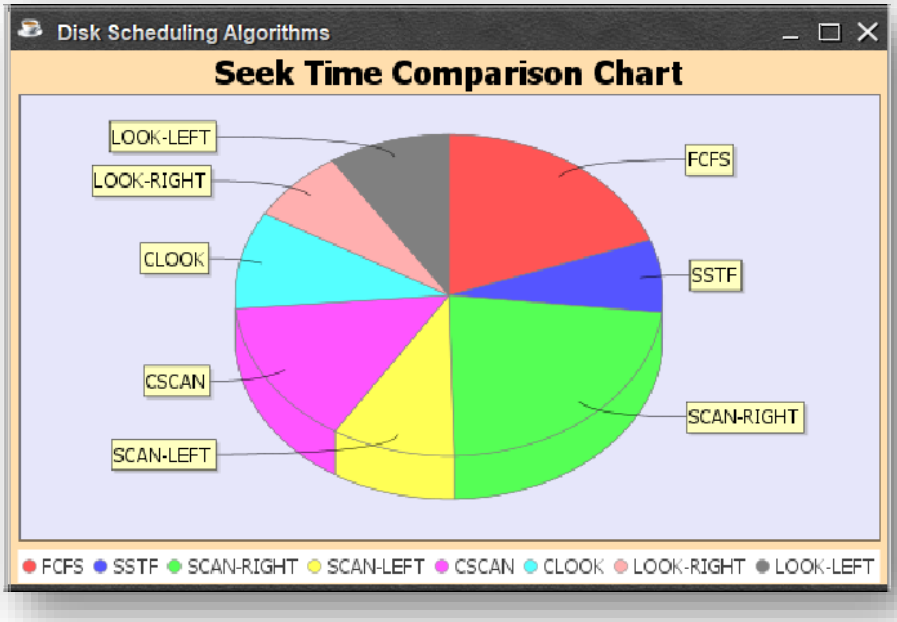
→ **CLOOK is selected the preface shows its small description All the tabs are required Previous header is used to determine the direction if greater than initial header the Left to Right and if smaller than left to right.**





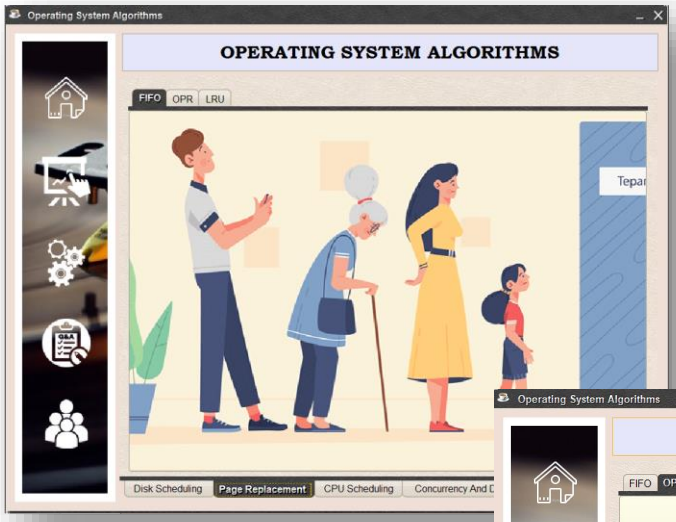
COMPARISION OF ALGORITHMS





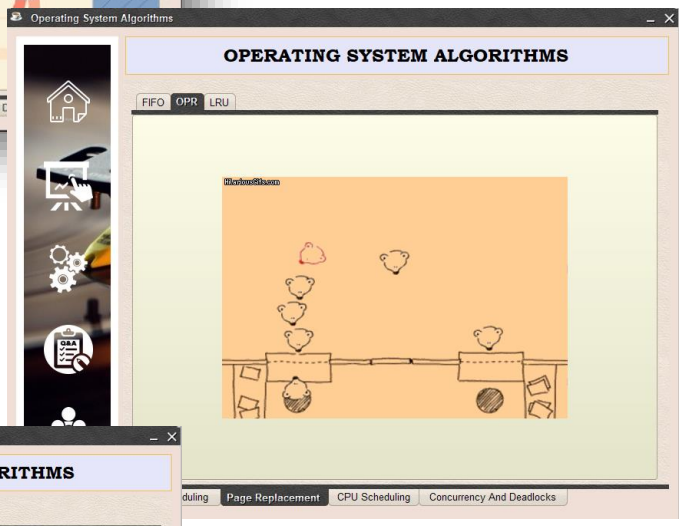
Algorithm implementation of Page replacement

→ Page Replacement Algorithm was implemented by our seniors which we have combined in our project. They implemented 3 algorithms of Page Replacement which are First In First Out, Least Recently Used and Optimal Page Replacement. Select the algorithm from the tabbed pane and click on the GIF which is linked to the respective algorithm implementation page.



FIRST INPUT FIRST OUTPUT

(FIFO)



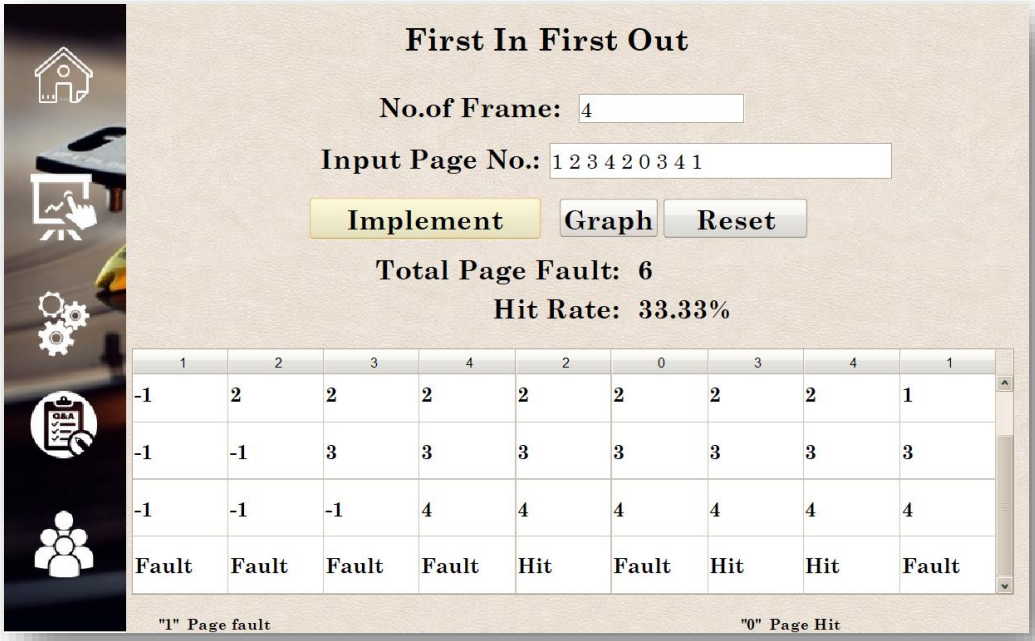
OPTIMAL PAGE REPLACEMENT

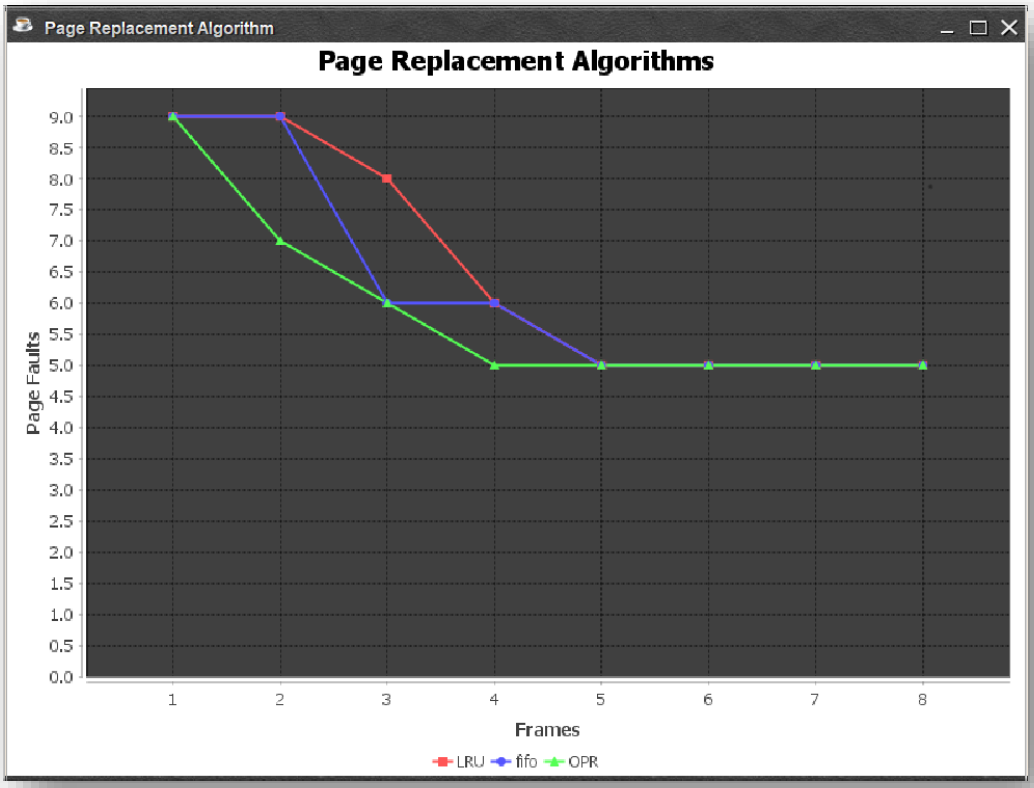
(OPR)

LEAST RECENTLY USED

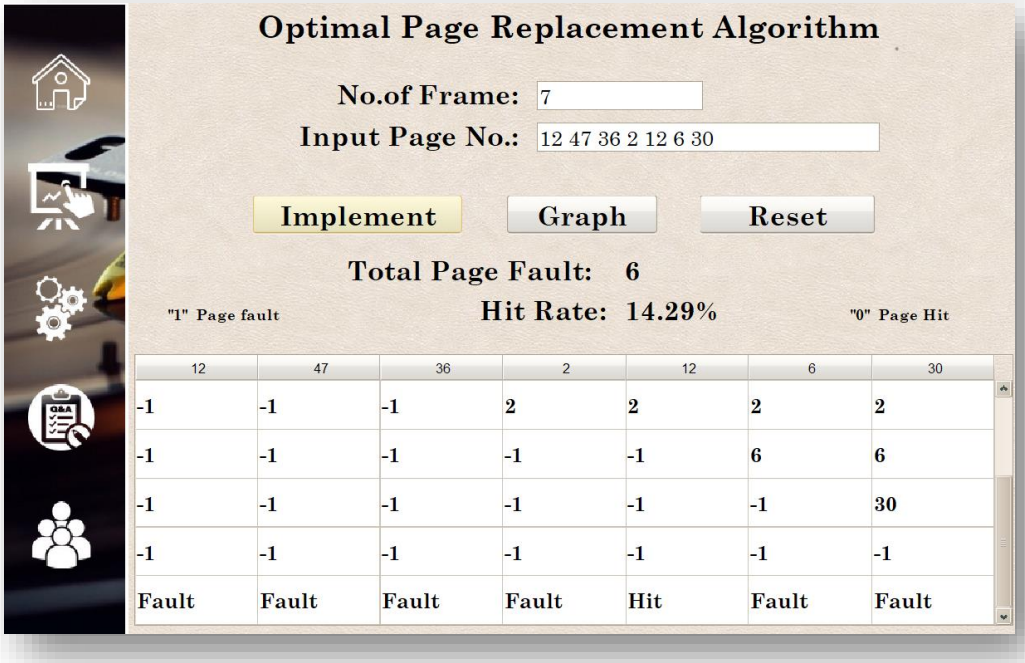
(LRU)

FIRST INPUT FIRST OUTPUT (FIFO)

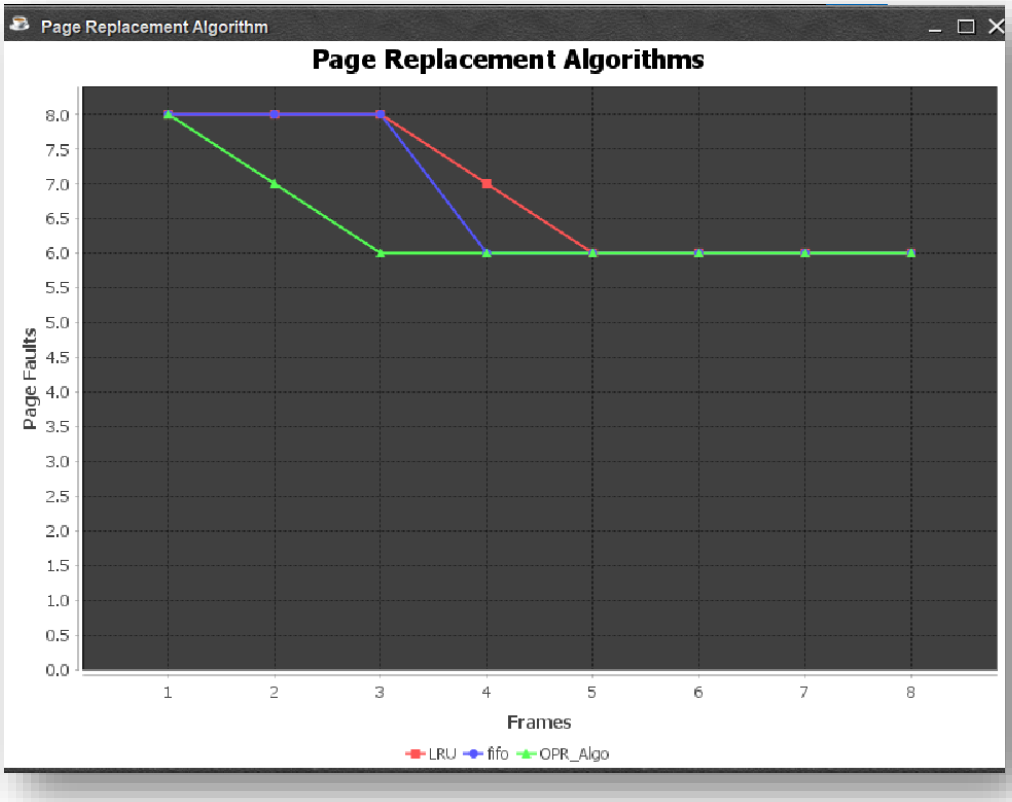




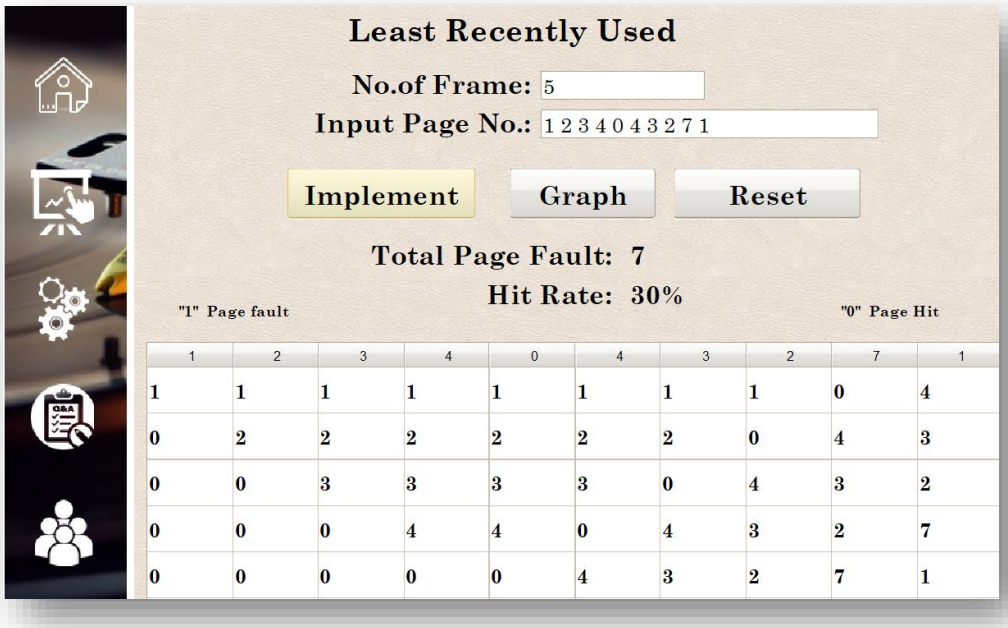
OPTIMAL PAGE REPLACEMENT (OPR)



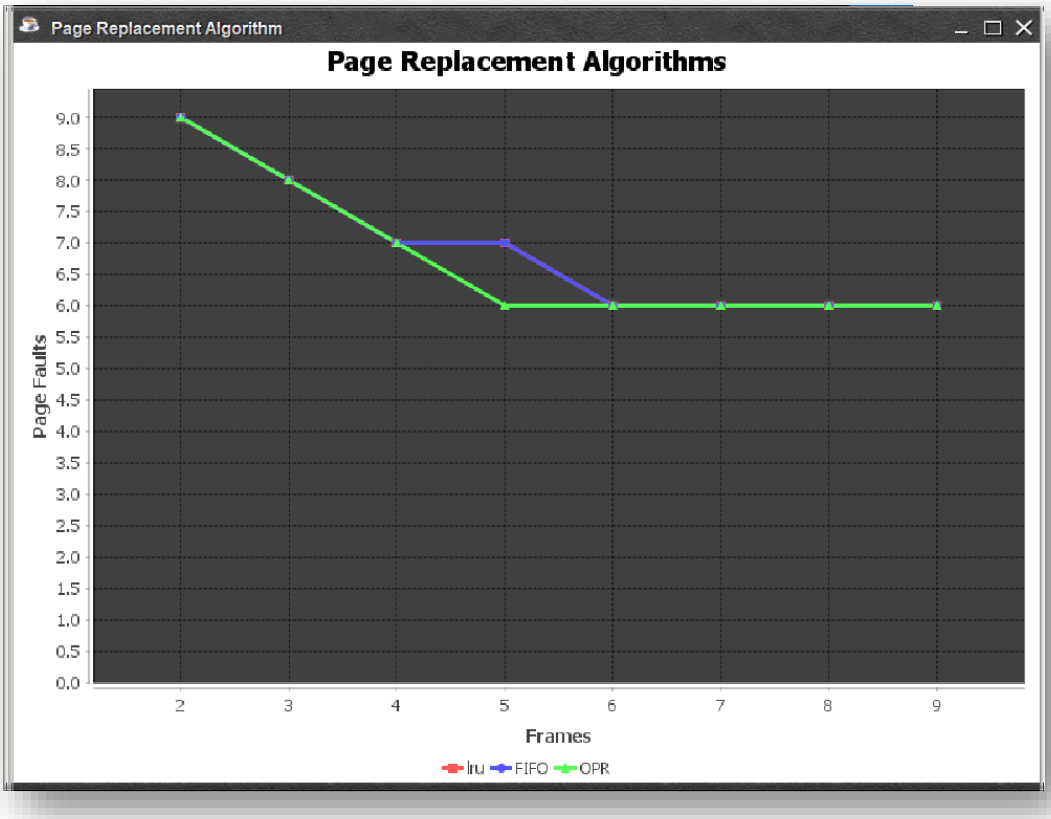
12	47	36	2	12	6	30
-1	-1	-1	2	2	2	2
-1	-1	-1	-1	-1	6	6
-1	-1	-1	-1	-1	-1	30
-1	-1	-1	-1	-1	-1	-1
Fault	Fault	Fault	Fault	Hit	Fault	Fault



Least Recently Used (LRU)



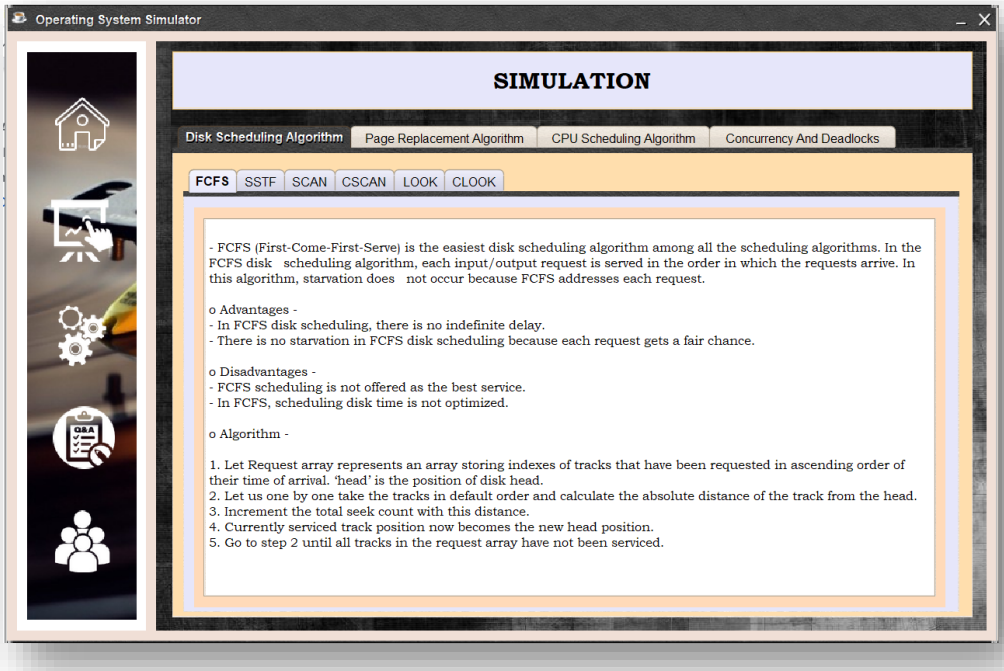
1	2	3	4	0	4	3	2	7	1
1	1	1	1	1	1	1	1	0	4
0	2	2	2	2	2	2	0	4	3
0	0	3	3	3	3	0	4	3	2
0	0	0	4	4	0	4	3	2	7
0	0	0	0	0	4	3	2	7	1

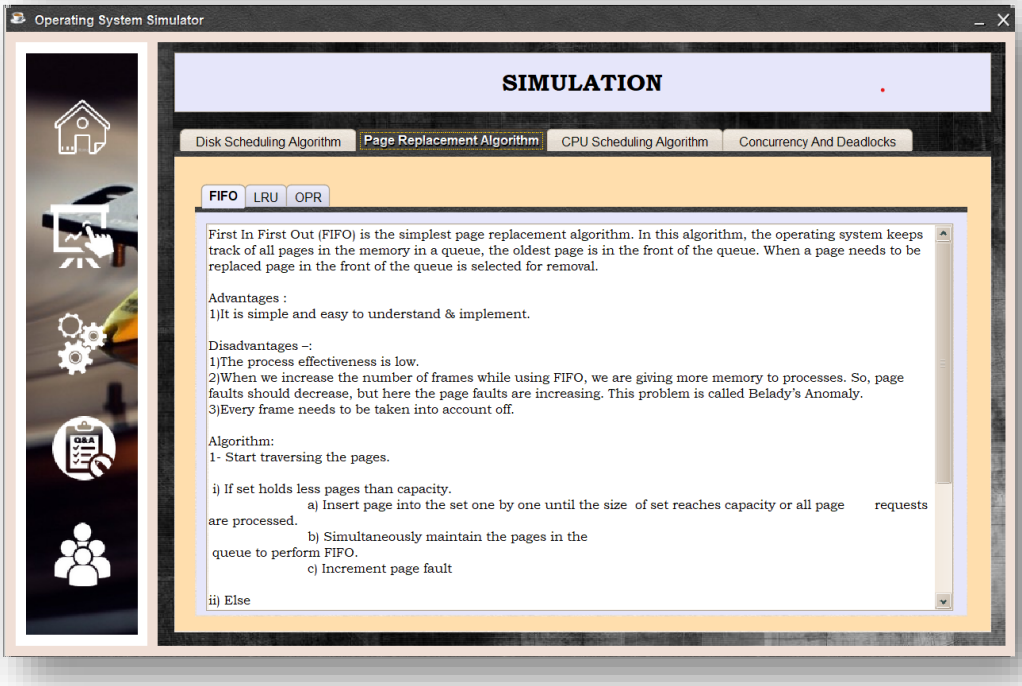


SIMULATION TAB

THE SIMULATOR TAB DISPLAYS:

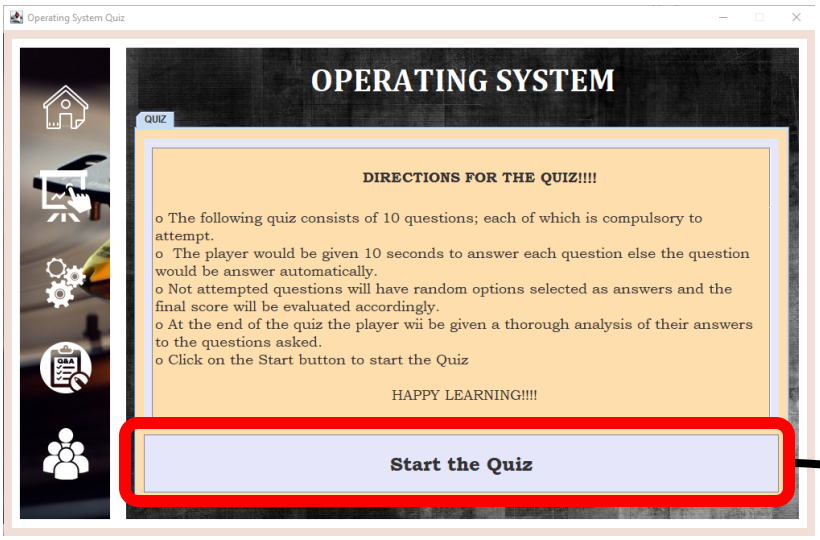
1. **Description of algorithm**
2. **Advantages and Disadvantages of respective algorithm**
3. **Implementation/steps for running an algorithm**



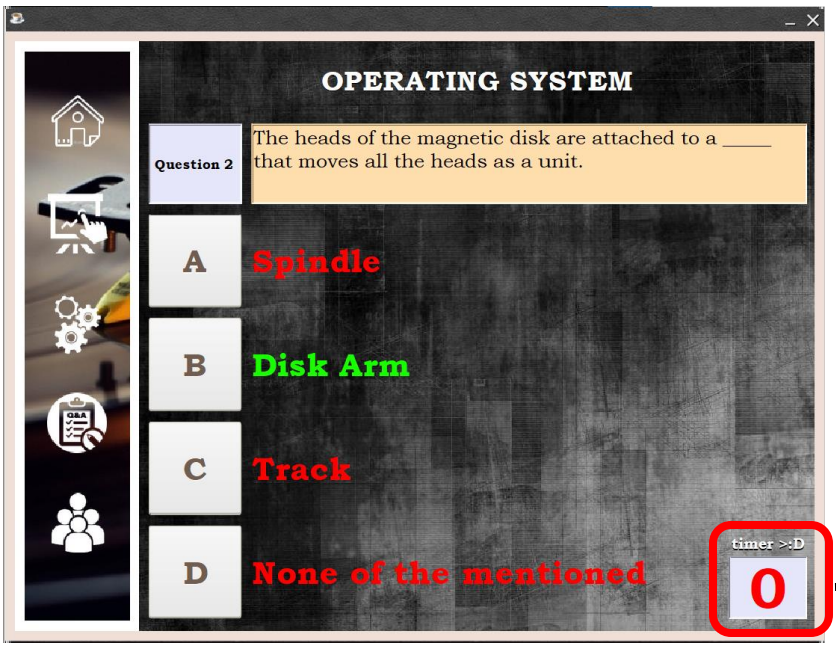


QUIZ TAB OF DESKTOP APP

We have introduced a new feature in our desktop app which is a QUIZ on Operating System.
The rules for the quiz are displayed before the starting of the quiz. On clicking **Start the Quiz** button a new frame is displayed with 1st question on it along with 4 options. On selecting any 4 options the correct one will be highlighted in green color and wrong options will be highlighted in red. For each question a time of ten seconds is given. This can be visually seen through the timer in the bottom right corner. At the end of quiz, the player is given a brief analysis of his answers.



Click on this button to start the quiz



10 SECONDS TIMER PER QUESTION

ABOUT US:

THIS FRAME CONSIST OF THE IMAGES AND NAMES OF ALL THE DEVELOPERS OF 2 GUI I.E. PAGE REPLACEMENT ALGORITHMS AND DISK SCHEDULING ALGORITHMS

CONCLUSION:

All the above algorithms and tabs are running properly with no errors in the code.
Through this project our team got the chance to learn following things given below:

- We learnt that with teamwork we can achieve our goal in limited time.
- We learnt how to implement different libraries of java and connect all the frames to each other for seamless user experience.
- We learned the concept of disk scheduling algorithm thoroughly.
- We learned to implement animation and Desktop App with video, charts, graphs, and content.

Hence, team 14 have implemented all the algorithms successfully within the given time frame and have submitted their video portraying the working of the code properly.
Thanking Chintan sir and Samir Sir for giving us this eminent opportunity to create this GUI and guiding us throughout out project. We are pleased to announce that we have completed our work and will be waiting for your suggestions and constant support in the future.

FUTURE WORK

For future work of implementing the other 2 algorithms we have put buttons and frames in our project dedicated to CPU Scheduling and Concurrency and Deadlock so that anyone can work on our project and implement these two algorithms. Hence the entire Operating System algorithms can be implemented in our GUI Application.