PROCESS MANAGEMENT SIMULATOR

A PROJECT REPORT

Submitted by

PARVATHY V NAIR [Reg No: RA2211003010295]

ANYESHA BISWAS [Reg No: RA2211003010298]

ACHANTA BHAVYA SREE [Reg No: RA2211003010316]

Under the Guidance of

DR. M.ELIAZER

Assistant Professor (Sr.Grade), Department of Computing Technology

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGY

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled "**PROCESS MANAGEMENT SIMULATOR**" is the bonafide work of Ms. Parvathy V Nair [Reg. No.: RA2211003010295] and Ms. Anyesha Biswas [Reg. No.RA2211003010298] and Ms. Achanta Bhavya Sree [Reg. No.: RA2211003010316] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on thebasis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

DR. M.ELIAZER
Assistant Professor
Department of Computing technology
SRM Institute of Science and Technology
Kattankulathur

TABLE OF CONTENTS

C.NO.	*TITLE*		*PAGE NO.*
	Abstract		ix
	List of Symbols and Abbreviations		xiii
1.	INTRODUCTION		
	1.1	General	
	1.2	Purpose	
	1.3	Scope	
	1.4	Artificial Intelligence, Machine Learning and Deep Learning	
	1.5	Computer Vision	
2	LITERATURE REVIEW		
	2.1	DriftNet	
	2.2	Previous Frameworks	
	2.3	ResNets	
	2.4	EfficientNets	
	2.5	RandAugment	
	2.6	Hausdorff Distance	
3	PROPOSED METHODOLOGY		
	3.1	Dataset	
	3.2	Feature Extractor	
	3.3	Classifier	
	3.4	Detection	
	3.5	Algorithm	
4	CODE		
5	RESULT		
6	CONCLUSION		

ABSTRACT

Our project endeavors to develop a comprehensive Process Scheduling Simulator with the capability to emulate diverse process scheduling algorithms commonly employed in operating systems. The simulator, driven by inputs like arrival time, burst time, and priority, will exhibit the sequential execution order of processes, aligning with the chosen scheduling algorithm.

This innovative tool will incorporate a range of algorithms, including but not limited to First-Come-First-Serve (FCFS), Shortest Job First (SJF), Round-Robin, Priority-Based, and others. By doing so, our solution aspires to furnish a valuable resource catering to users at varying proficiency levels—ranging from beginners seeking foundational understanding to advanced users aiming for in-depth analysis and optimization of process scheduling algorithms.

Our project aims to create a Process Scheduling Simulator that can simulate different process scheduling algorithms used in operating systems. The simulator will take inputs such as arrival time, burst time, and priority of the process and show the execution order of the processes based on the selected scheduling algorithm.

The simulator will implement various algorithms such as First-Come-First-Serve (FCFS), Shortest Job First (SJF), Round-Robin, Priority-Based, and others. Our solution aims to provide a useful tool for learning, analyzing, and optimizing process scheduling algorithms from beginner to advanced.

LIST OF SYMBOLS AND ABBREVIATIONS

1. **Processes:**
   - P1, P2, P3, ...: Process identifiers or names.
   - PCB: Process Control Block, representing information about a process.
   - ➝ or →: Indicates process execution or transition.

2. **Queues:**
   - Ready Queue: Queue of processes ready to execute.
   - Blocked Queue: Queue of processes waiting for an event (e.g., I/O).

3. **Resources:**
   - CPU: Central Processing Unit.
   - I/O: Input/Output.
   - RAM: Random Access Memory.

4. **Scheduling:**
   - FCFS: First-Come-First-Serve scheduling algorithm.
   - SJF: Shortest Job First scheduling algorithm.
   - RR: Round Robin scheduling algorithm.

5. **States:**
   - New: Process in the initial state.
   - Ready: Process waiting to be executed.
   - Running: Process currently being executed.
   - Blocked: Process waiting for an event (e.g., I/O).

6. **Arrows and Lines:**
   - ➞ or →: Transition or flow of control.
   - ↘ or ↙: Indicates a process moving from a running state to a blocked state.
   - ↗ or ↖: Indicates a process moving from a blocked state to a ready state.

7. Semaphore:

- S: Semaphore used for process synchronization.

8. Clock Time:

- T: Time unit or clock cycle.

9. Input/Output:

- I/O Request: Symbol representing a request for input/output.

10. Error/Exception:

- ⚠: Indicates an error or exception in the system.

11. Concurrency:

- Concurrent Processes: Symbol indicating processes running simultaneously.

12. Communication:

- IPC: Inter-Process Communication.

13. Termination:

- ⛔: Indicates process termination or system shutdown.

14. Priority:

- High Priority: Symbol indicating a high-priority process.

15. Memory:

- MMU: Memory Management Unit.

17. Message Passing:

- Message: Symbol representing communication between processes.

18. Quantum Time:

- Quantum: Time quantum in a Round Robin scheduling system.

1. INTRODUCTION

1.1 GENERAL :

In an operating system, process management OS (Operating System) refers to the set of activities involved in creating, scheduling, and terminating processes.
A process management OS can be thought of as an instance of a program that is currently executing on a computer system.

1.2 PURPOSE :

Understanding the various process scheduling algorithms used in operating systems can be challenging for students and beginners.
It can be hard to visualize how different algorithms prioritize and allocate resources to processes, which can affect system performance.

1.3 SCOPE :

Managing processes is an important aspect of any modern operating system, as it allows multiple programs to run concurrently and share system resources efficiently.
The process control block (PCB) is a data structure used by the operating system to manage each process. It contains all the information needed to manage the process, including the process ID, program counter, register values, memory information, and process state.

1.4 ARTIFICIAL INTELLIGENCE , MACHINE LEARNING AND DEEP LEARNING :

Artificial Intelligence (AI) enhances process management simulators in operating systems by optimizing resource allocation, predicting system behavior, and automating decision-making. Through machine learning algorithms, AI adapts to dynamic workloads, improving scheduling efficiency and minimizing latency. It enables predictive analysis of process states, aiding in proactive issue resolution. AI-driven simulators facilitate a more realistic emulation of complex operating system scenarios, aiding developers and administrators in designing robust and adaptive systems.

Machine learning plays a pivotal role in enhancing process management simulators in operating systems by enabling intelligent decision-making. Algorithms can analyze historical data to optimize task scheduling, predict resource demands, and dynamically adapt to varying workloads. This leads to improved efficiency, better resource utilization, and enhanced responsiveness in handling diverse computing tasks. By learning from patterns and adapting in real-time, machine learning contributes to the agility and effectiveness of process management in operating system simulations.

Deep Learning plays a crucial role in enhancing process management simulators in operating systems by enabling intelligent decision-making and adaptive system behavior. Through neural networks, it can learn and optimize scheduling algorithms, predict resource demands, and enhance real-time process allocation. This leads to improved system efficiency, resource utilization, and responsiveness, contributing to a more dynamic and intelligent operating environment.

1.5 COMPUTER VISION :

Computer Vision in a process management simulator enhances operating system monitoring by visually interpreting real-time data. It allows for intuitive representation of processes, resource utilization, and system states through image analysis. By translating complex system dynamics into visual cues, Computer Vision aids administrators in identifying bottlenecks, inefficiencies, and potential improvements. This visual insight facilitates efficient decision-making, leading to optimized resource allocation and improved overall system performance in operating system process management.

2. LITERATURE REVIEW :

2.1 DRIFTNET :

The Operating System Simulation (OSS) system is an operating system simulation tool to simulate the multiprocessing operating system which contains six major components, which are process management, scheduler, memory management, message management, semaphore management and resource management [1]. OSS is used to get the clear understanding of how an operating system works and to study the impact of various algorithms on the performance of an operating system. Operating systems provide many functions in which the process management is one of the most important. For the process management, various types of scheduling algorithms are used. The scheduling algorithm’s main goals include best CPU utilization, Throughput, T.T., Waiting Time (W.T.), Response Time (R.T.) and threads fairness. In the last thirty years there has been a huge amount of research in the area of disk scheduling [2]. The main objective was to design the scheduling algorithms with some verifiable properties. The CPU switches among processes in the multi programming environment and this is achieved by the use of CPU scheduling techniques/algorithms. The OS should allow processes as much as possible to run all time so the CPU utilization can be maximized. Thus a number of processes are kept in memory concurrently and each process is selected by the OS so that it can occupy the CPU.

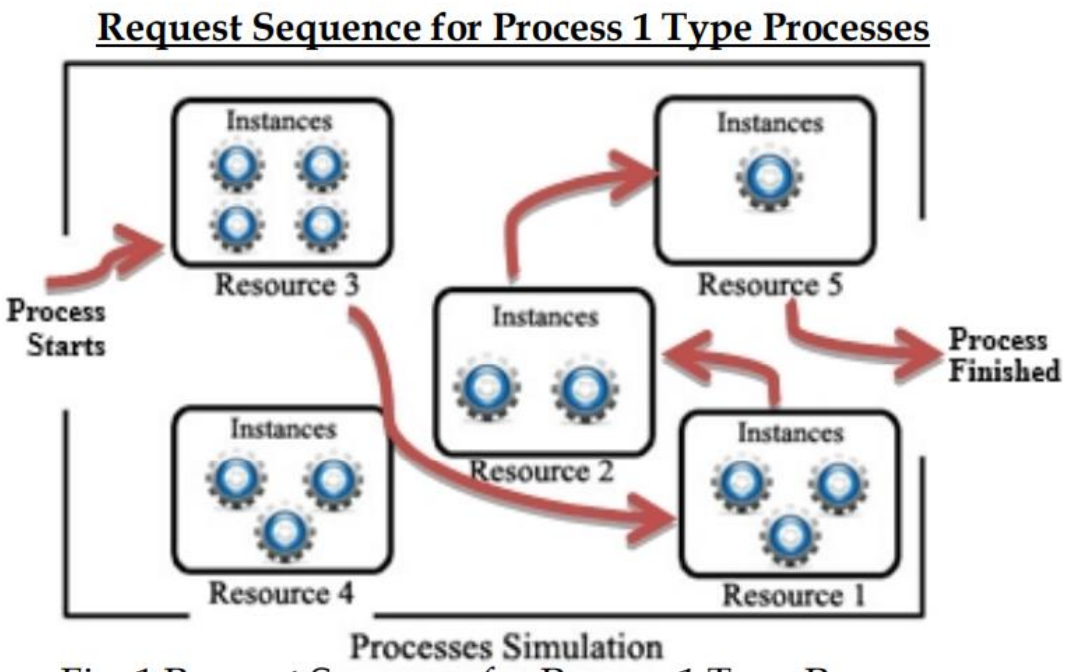
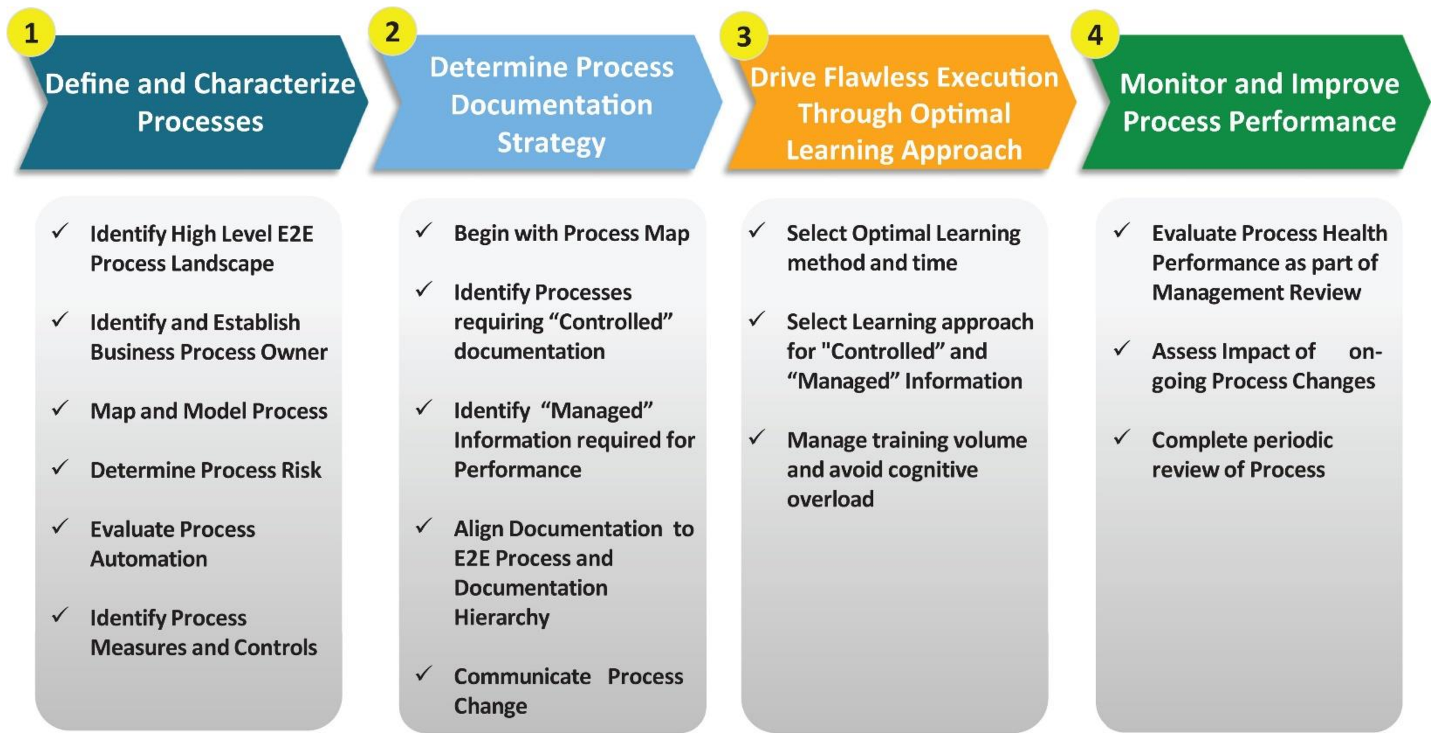


Fig. 1 Request Sequence for Process 1 Type Processes

2.2 PREVIOUS FRAMEWORKS :

A process management framework ensures processes are identified, defined, and implemented in a common way to sustain and *improve* clinical development activities, quality objectives, and deliverables. This framework outlines a potential “roadmap” for implementing processes and allows for an optimal processes design to avoid gaps and overlaps. Figure 2 depicts a conceptual framework or methodology for process implementation. The steps described in the process management framework are applicable to organizations of all sizes; however, they may need to be scaled to accommodate the size of the organization. In May 2017, a voluntary anonymous survey of 19 TransCelerate member companies indicated that 80% of member companies have started or are well underway in the implementation of a process-driven approach. However, member companies cited challenges associated with successful implementation. The primary challenges were (1) lack of leadership commitment, understanding, and change management and (2) lack of defined methodology or framework for implementing processes.


1
Define and Characterize Processes
✓ Identify High Level E2E Process Landscape
✓ Identify and Establish Business Process Owner
✓ Map and Model Process
✓ Determine Process Risk
✓ Evaluate Process Automation
✓ Identify Process Measures and Controls
2
Determine Process Documentation Strategy
✓ Begin with Process Map
✓ Identify Processes requiring “Controlled” documentation
✓ Identify “Managed” Information required for Performance
✓ Align Documentation to E2E Process and Documentation Hierarchy
✓ Communicate Process Change
3
Drive Flawless Execution Through Optimal Learning Approach
✓ Select Optimal Learning method and time
✓ Select Learning approach for "Controlled" and “Managed” Information
✓ Manage training volume and avoid cognitive overload
4
Monitor and Improve Process Performance
✓ Evaluate Process Health Performance as part of Management Review
✓ Assess Impact of on-going Process Changes
✓ Complete periodic review of Process

2.3 RESNETS :

Resnets are made by stacking these residual blocks together. The approach behind this network is instead of layers learning the underlying mapping, we allow the network to fit the residual mapping.

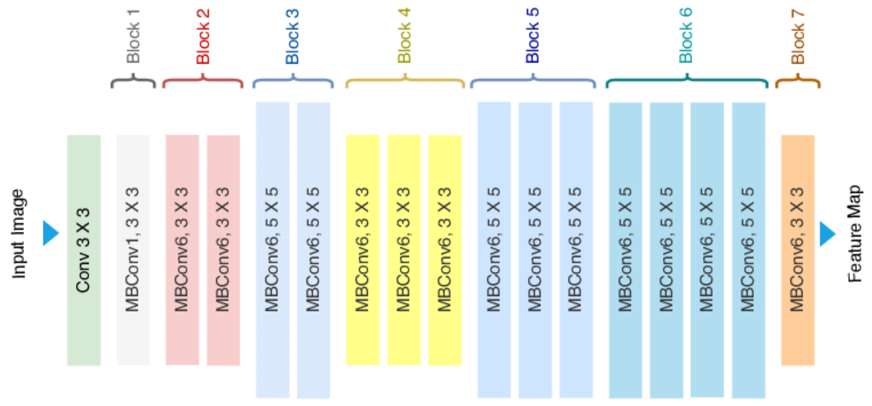
APQC's Seven Tenets of Process Management encapsulate the common characteristics exhibited by **successful process-managed organizations**. By assessing an organization's current state using the seven tenets, leaders can better discern which activities and strategic decisions will move the enterprise toward more mature and **streamlined process management practices**.

Once the assessment is complete, you can determine what basic changes can be made in each of the seven areas that will drive performance and ultimately increase the organization's **process maturity**. By considering all seven tenets, leaders can feel confident that process management plans address the organization holistically and will support a smooth and efficient transition from the old ways of doing things to the new.

2.4 EFFICIENTNETS :

EfficientNet is a convolutional neural network built upon a concept called "compound scaling." This concept addresses the longstanding trade-off between model size, accuracy, and computational efficiency. The idea behind compound scaling is to scale three essential dimensions of a neural network: width, depth, and resolution.

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

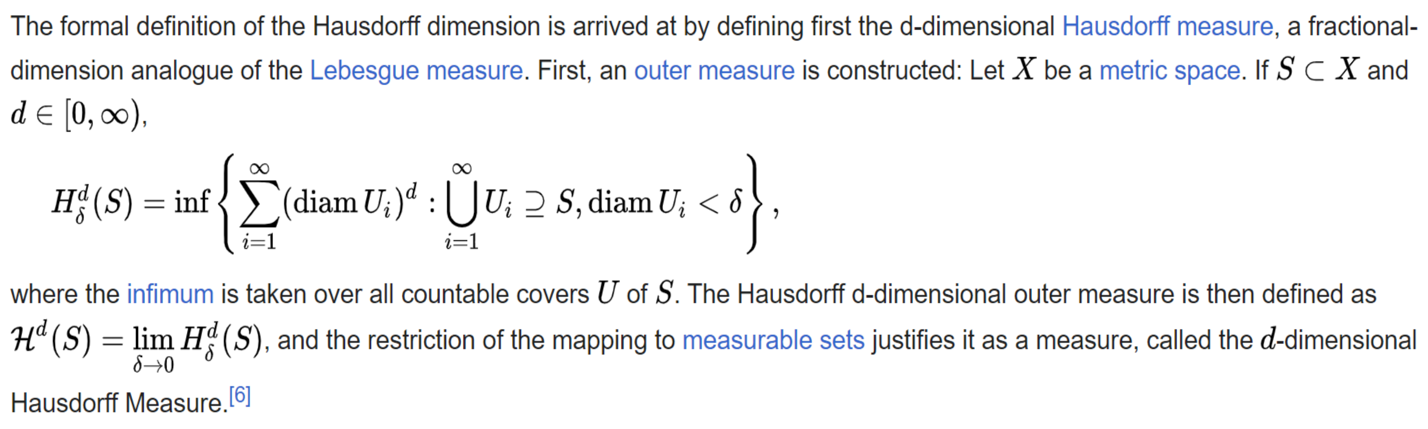


2.5 RANDAUGMENT :

RandAugment is an automated data augmentation method. The search space for data augmentation has 2 interpretable hyperparameter N and M. N is the number of augmentation transformations to apply sequentially, and M is the magnitude for all the transformations. To reduce the parameter space but still maintain image diversity, learned policies and probabilities for applying each transformation are replaced with a parameter-free procedure of always selecting a transformation with uniform probability 1/K. Here K is the number of transformation options. So given N transformations for a training image, RandAugment may thus express KN potential policies. That's where the Business Process Management-Discipline (BPM-Discipline) helps. The BPM-Discipline has become the "Strategy Execution Engine" of an organization. It delivers significant business value by converting strategy into people and IT based execution at pace with certainty. The BPM-Discipline is implemented through the "process of process management". Organizations look for a way to systematically establish their process of process management fast and cost effectively. This can be achieved using a holistic framework and reference model. This whitepaper introduces the BPM-Discipline and how it is implemented through the process of process management.

2.6 HAUSDORFF DISTANCE :

Hausdorff distance (HD) has been an indispensable tool to address computer vision and pattern recognition problems. Compared with other metrics, HD incorporates critical details of objects (position and shapes) to output reasonable distances between pairs of point sets. Despite its capabilities, HD is highly sensitive to outliers and noisy data. From previous studies, we discovered that the issues have been inadequately addressed. In the present work, a technique is proposed to empower HD under undesirable conditions of data.



The formal definition of the Hausdorff dimension is arrived at by defining first the d-dimensional Hausdorff measure, a fractional-dimension analogue of the Lebesgue measure. First, an outer measure is constructed: Let X be a metric space. If $S \subset X$ and $d \in [0, \infty)$,

$$H_\delta^d(S) = \inf \left\{ \sum_{i=1}^{\infty} (\operatorname{diam} U_i)^d : \bigcup_{i=1}^{\infty} U_i \supseteq S, \operatorname{diam} U_i < \delta \right\},$$

where the infimum is taken over all countable covers U of S. The Hausdorff d-dimensional outer measure is then defined as $\mathcal{H}^d(S) = \lim_{\delta \to 0} H_\delta^d(S)$, and the restriction of the mapping to measurable sets justifies it as a measure, called the d-dimensional Hausdorff Measure.[6]

3. PROPOSED METHODOLOGY

The proposed methodology supports SPI based on process and project management alignment. Process and project alignment is defined as the degree to which project goals and plans support and are supported by the process practices. Moreover, it involves a real match between process practices and project activities, work products and actors. However, several modifications in a project can cause misalignments with the development process. These modifications can be management innovations or changes in the way activities are executed. Furthermore, a modification may regard not only the considered activity, work product or actor, but it could also affect other elements having a dependence relation with the modified one.

3.1 DATASET :

There are various datasets available for process management, depending on the specific aspect of process management you're interested in. Process management can refer to a wide range of activities, from business processes to manufacturing processes.

1. Business Process Management (BPM) Datasets:

- [BPI Challenge](https://www.win.tue.nl/bpi/)
- [UCI Machine Learning Repository - Business Process Management Data](http://archive.ics.uci.edu/ml/datasets/Business+Process+Management+Data)

2. Manufacturing Process Datasets:

- [SECOM Manufacturing Data](https://archive.ics.uci.edu/ml/datasets/SECOM)
- [Condition Monitoring of Hydraulic Systems Data Set](https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems)

3. Workflow Management Datasets:

- [ProM Datasets](http://www.promtools.org/doku.php?id=prom6:datasets)

4. Healthcare Process Datasets:

- [MIMIC-III](https://physionet.org/content/mimiciii/1.4/)

5. Service Process Datasets:

- [Service Process Discovery and Analysis Challenge (SPDAC)](https://www.spdac.com/)

3.2 FEATURE EXTRACTOR :

1. Time-based Features:

- Duration: Time taken for the completion of a process or a specific task within a process.
- Timestamps: Temporal information indicating when certain events occurred.

2. Event Log Features:

- Frequency of Events: Count of how often specific events occur in a process.
- Sequence Patterns: Patterns of events or activities in a sequence.

3. Resource Utilization Features:

- Resource Allocation: How resources (human or machine) are allocated during the process.
- Concurrency: The number of tasks or processes happening concurrently.

4. Performance Metrics:

- Throughput: The number of processes or tasks completed per unit of time.
- Efficiency: Ratio of completed tasks to the total resources consumed.

5. Variability Measures:

- Standard Deviation: Measure of how spread out the process data is.
- Variation Coefficient: Ratio of the standard deviation to the mean.

6. Text Mining Features (if dealing with textual data):

- Natural Language Processing (NLP) Features: Extracting information from text data, such as sentiment, key phrases, etc.

7. Machine Learning-Driven Features:

- Predicted Labels: If using predictive models, features derived from model predictions.
- Anomaly Scores: If detecting anomalies, features related to the likelihood of an event being anomalous.

8. Graph-based Features (if dealing with processes represented as graphs):

- Node and Edge Characteristics: Properties of nodes and edges in the process graph.

3.3 CLASSIFIER :

1. Decision Trees:

- Decision trees are used to make decisions based on a series of questions. They are interpretable and can handle both numerical and categorical data.

2. Support Vector Machines (SVM):

- SVM is used for both classification and regression tasks. It works well in high-dimensional spaces and is effective when there is a clear margin of separation between classes.

3. Naive Bayes:

- Naive Bayes classifiers are based on Bayes' theorem and are particularly useful when dealing with text classification or situations where features are conditionally independent.

4. Neural Networks:

- Deep learning models, such as neural networks, can be used for complex tasks and can automatically learn hierarchical representations of data. They are especially useful when dealing with large amounts of data.

5. K-Nearest Neighbors (KNN):

- KNN classifies instances based on the majority class of their k-nearest neighbors in the feature space. It is simple but can be computationally expensive for large datasets.

6. Ensemble Methods (e.g., AdaBoost):

- Ensemble methods combine multiple weak learners to create a stronger model. AdaBoost, for example, iteratively gives more weight to misclassified instances.

7. Gradient Boosting Machines (e.g., XGBoost, LightGBM):

- Gradient boosting methods build a series of weak learners sequentially, with each one trying to correct the errors of the previous one. XGBoost and LightGBM are popular implementations.

8. Rule-based Classifiers (e.g., C4.5, RIPPER):

- Rule-based classifiers generate a set of rules that collectively classify instances. C4.5 and RIPPER are examples of such classifiers.

3.4 DETECTION :

1. Anomaly Detection in Processes:

- Description: Identifying unusual patterns or behaviors in a process that deviate from the norm.
- Approach: Use statistical methods, machine learning algorithms, or rule-based systems to detect anomalies in process data.

2. Process State Detection:

- Description: Determining the current state or phase of a process.
- Approach: Utilize process mining techniques, state machines, or machine learning models trained on labeled data to classify the current state of a process based on observed events or features.

3. Compliance Detection:

- Description: Ensuring that processes adhere to predefined rules, regulations, or standards.
- Approach: Use rule-based systems, logic-based reasoning, or machine learning models to evaluate whether observed process instances comply with specified rules and constraints.

4. Bottleneck Detection:

- Description: Identifying stages or components in a process that slow down the overall efficiency.
- Approach: Analyze process data to identify areas where tasks or resources are causing delays. Key performance indicators (KPIs) such as throughput and resource utilization can be used.

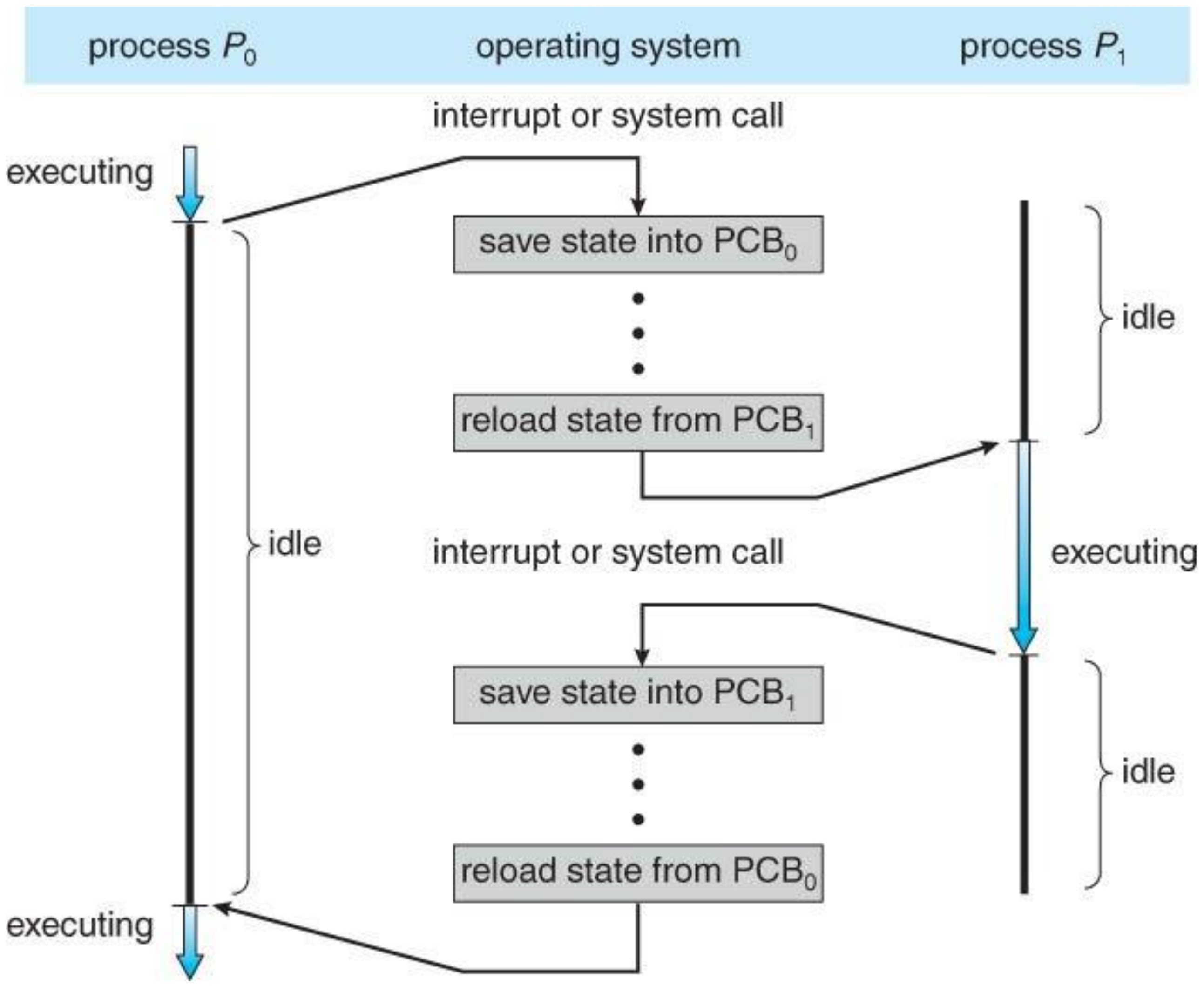
5. Predictive Maintenance in Processes:

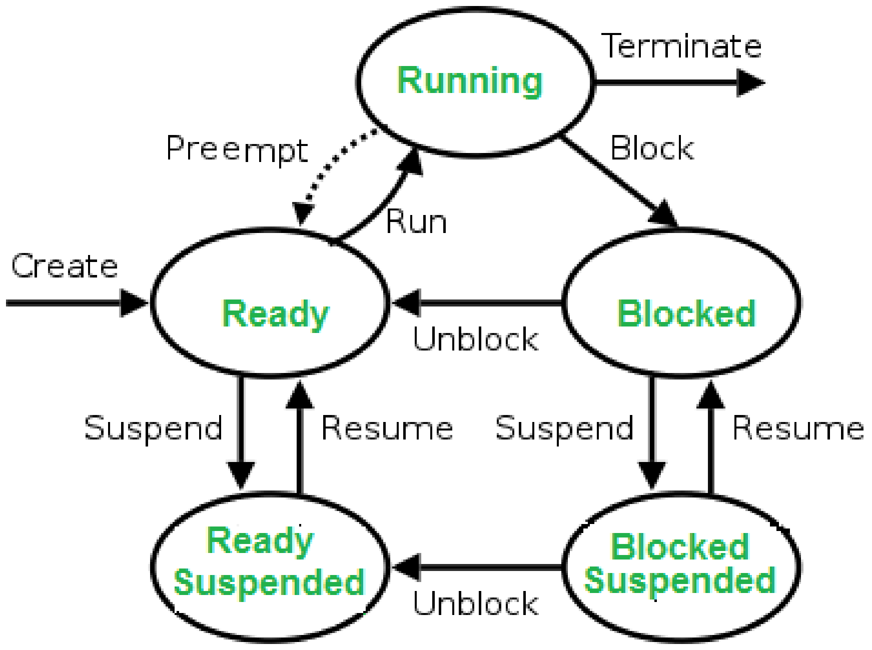
- Description: Anticipating and detecting potential issues in a process to perform maintenance proactively.
- Approach: Use predictive modeling, machine learning algorithms, or statistical methods to predict when certain components of a process are likely to fail or require maintenance based on historical data.

6. Change Detection in Processes:

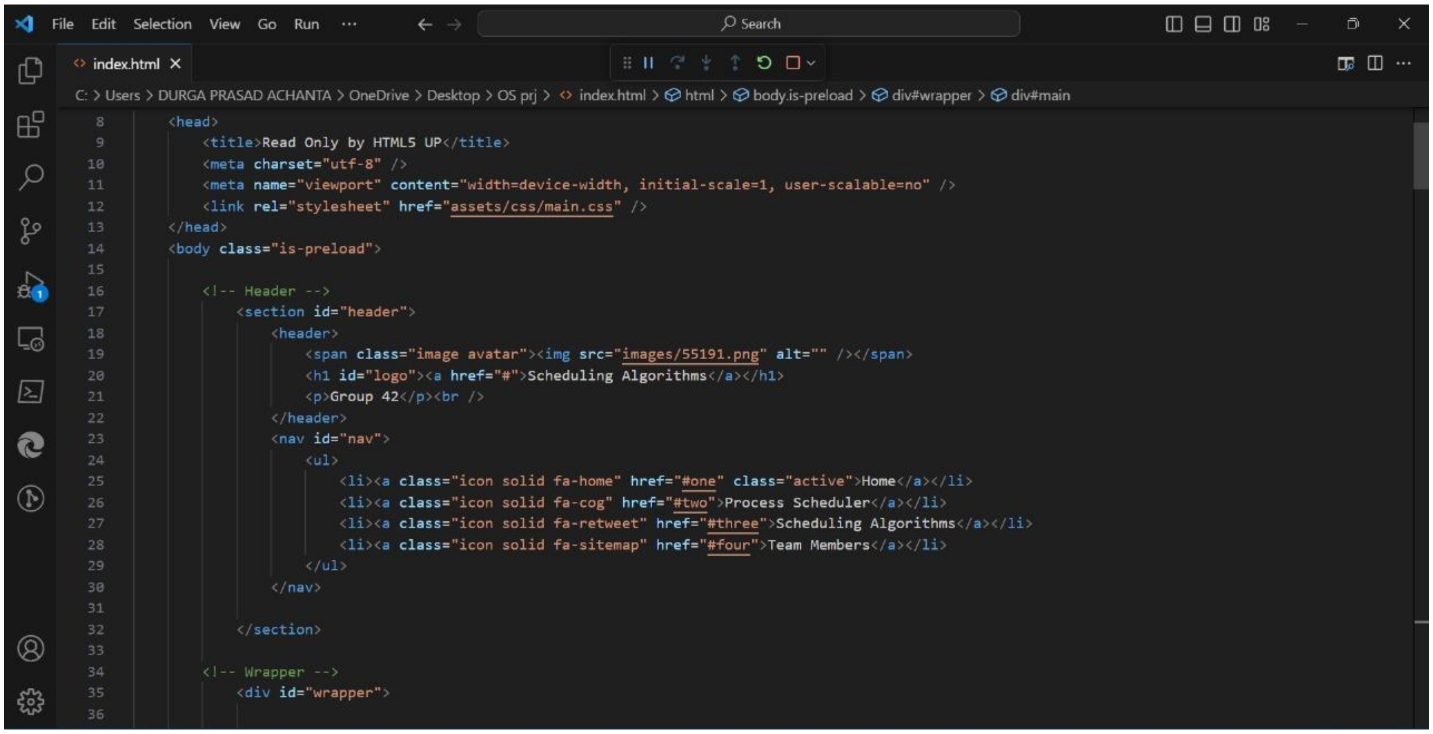
- Description: Identifying changes in the structure or behavior of a process over time.
- Approach: Utilize process mining techniques, change-point detection algorithms, or machine learning models to detect shifts in process dynamics or deviations from a baseline.

3.5 ALGORITHM :



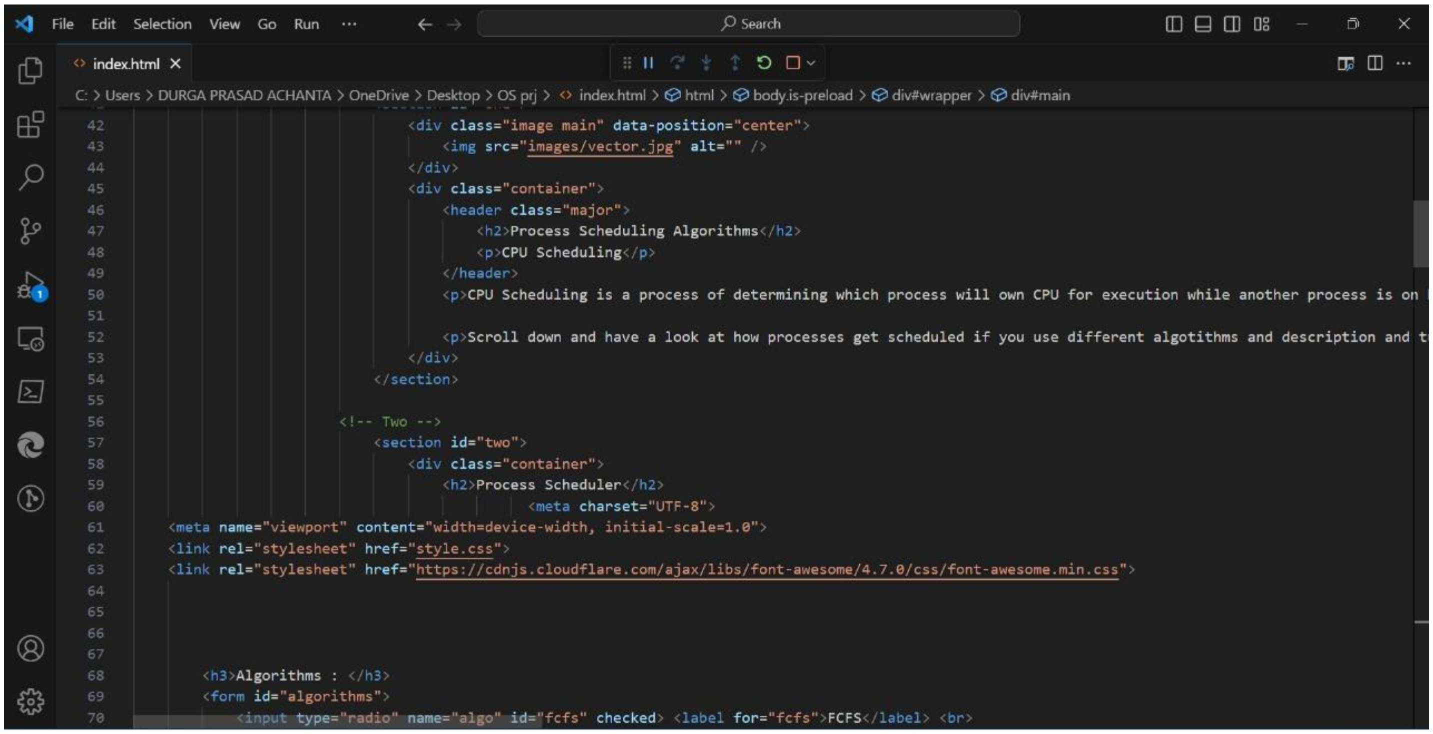


4. CODE



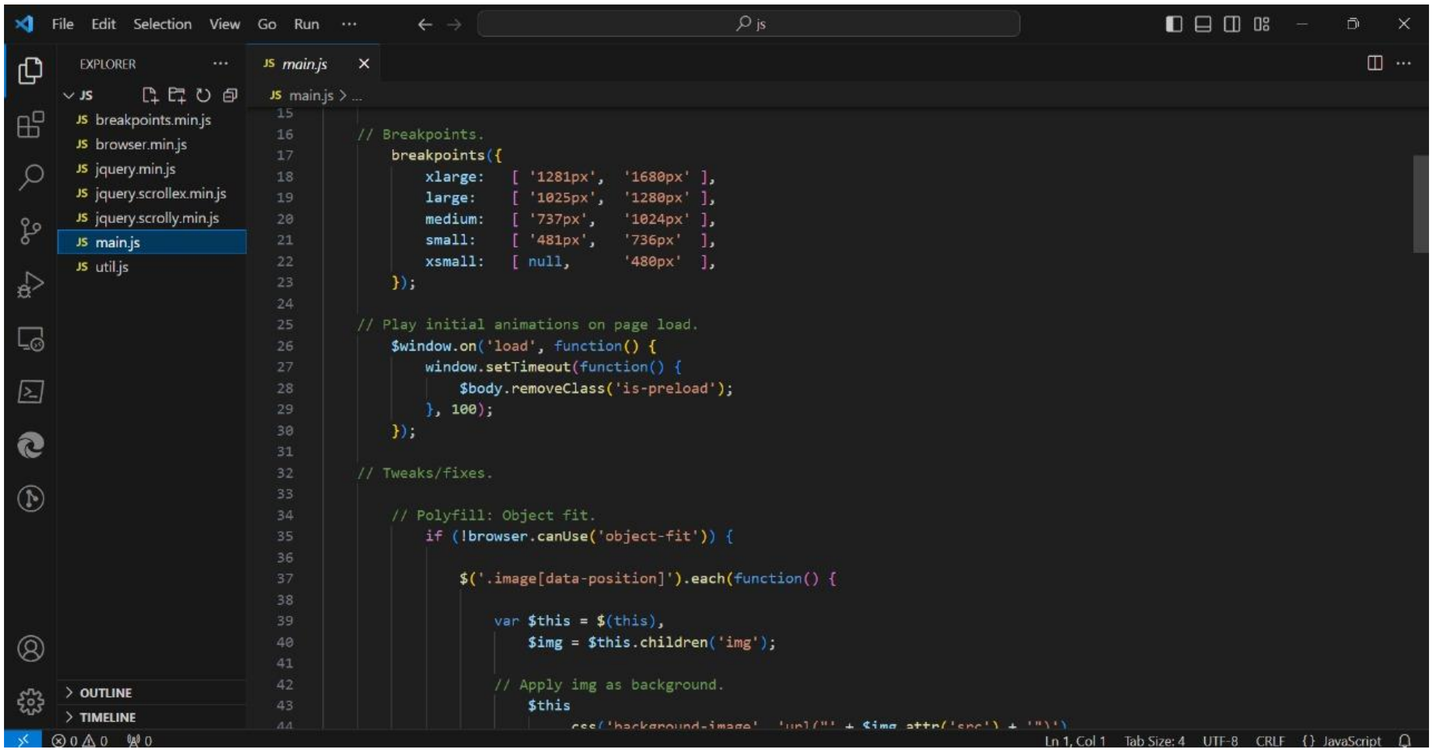
C: > Users > DURGA PRASAD ACHANTA > OneDrive > Desktop > OS prj > index.html > html > body.is-preload > div#wrapper > div#main

```html
    <head>
        <title>Read Only by HTML5 UP</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
        <link rel="stylesheet" href="assets/css/main.css" />
    </head>
    <body class="is-preload">

        <!-- Header -->
            <section id="header">
                <header>
                    <span class="image avatar"><img src="images/55191.png" alt="" /></span>
                    <h1 id="logo"><a href="#">Scheduling Algorithms</a></h1>
                    <p>Group 42</p><br />
                </header>
                <nav id="nav">
                    <ul>
                        <li><a class="icon solid fa-home" href="#one" class="active">Home</a></li>
                        <li><a class="icon solid fa-cog" href="#two">Process Scheduler</a></li>
                        <li><a class="icon solid fa-retweet" href="#three">Scheduling Algorithms</a></li>
                        <li><a class="icon solid fa-sitemap" href="#four">Team Members</a></li>
                    </ul>
                </nav>

            </section>

        <!-- Wrapper -->
            <div id="wrapper">

```

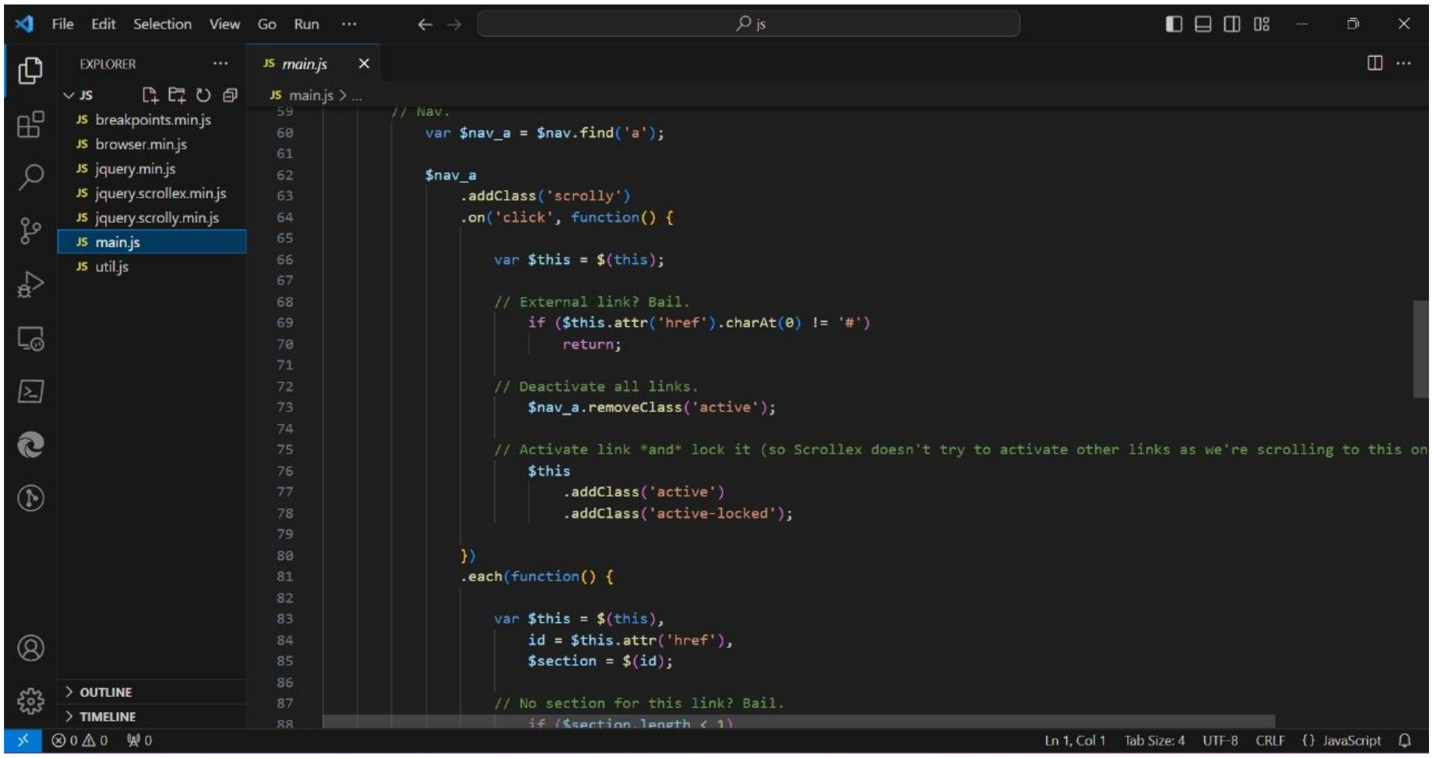


C: > Users > DURGA PRASAD ACHANTA > OneDrive > Desktop > OS prj > index.html > html > body.is-preload > div#wrapper > div#main

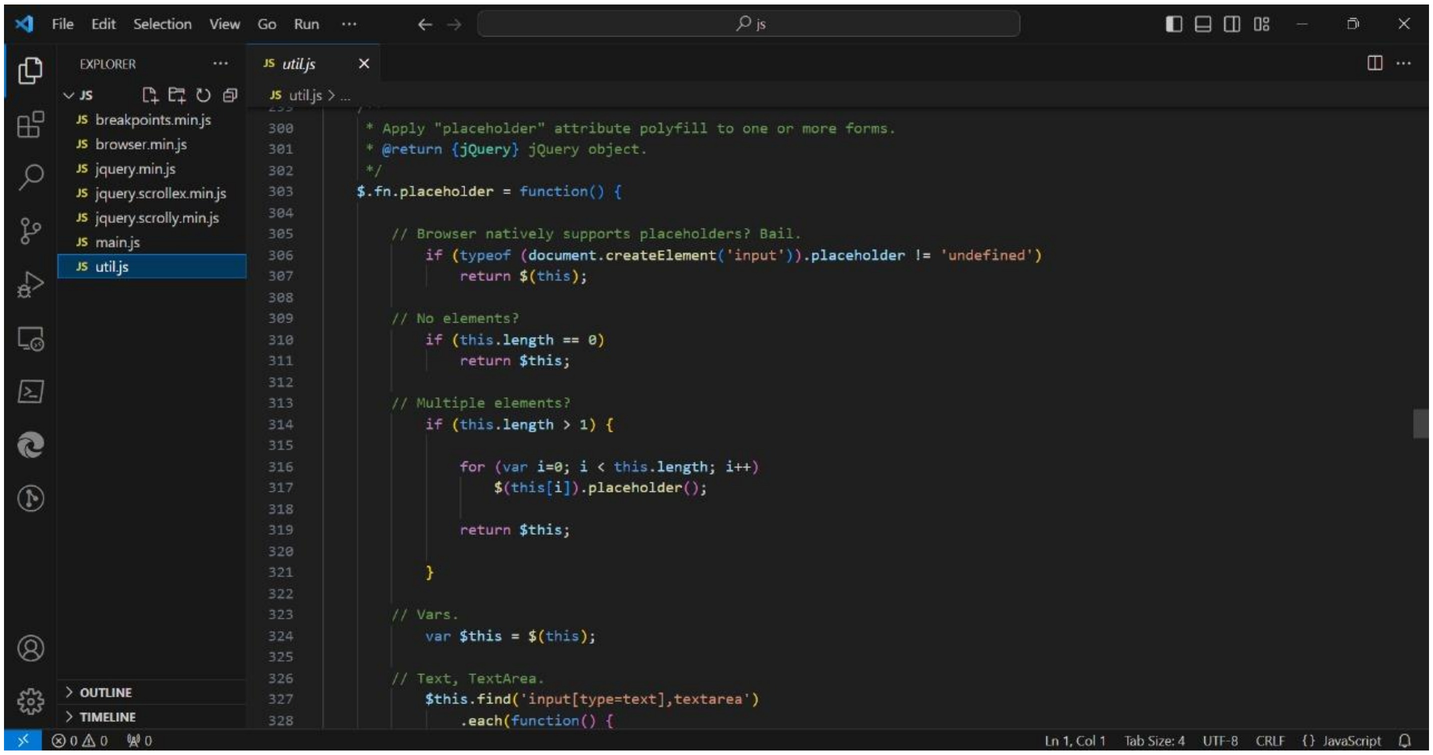
```html
                            <div class="image main" data-position="center">
                                <img src="images/vector.jpg" alt="" />
                            </div>
                            <div class="container">
                                <header class="major">
                                    <h2>Process Scheduling Algorithms</h2>
                                    <p>CPU Scheduling</p>
                                </header>
                                <p>CPU Scheduling is a process of determining which process will own CPU for execution while another process is on

                                <p>Scroll down and have a look at how processes get scheduled if you use different algotithms and description and t
                            </div>
                        </section>

                    <!-- Two -->
                        <section id="two">
                            <div class="container">
                                <h2>Process Scheduler</h2>
                                         <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">




        <h3>Algorithms : </h3>
        <form id="algorithms">
            <input type="radio" name="algo" id="fcfs" checked> <label for="fcfs">FCFS</label> <br>
```



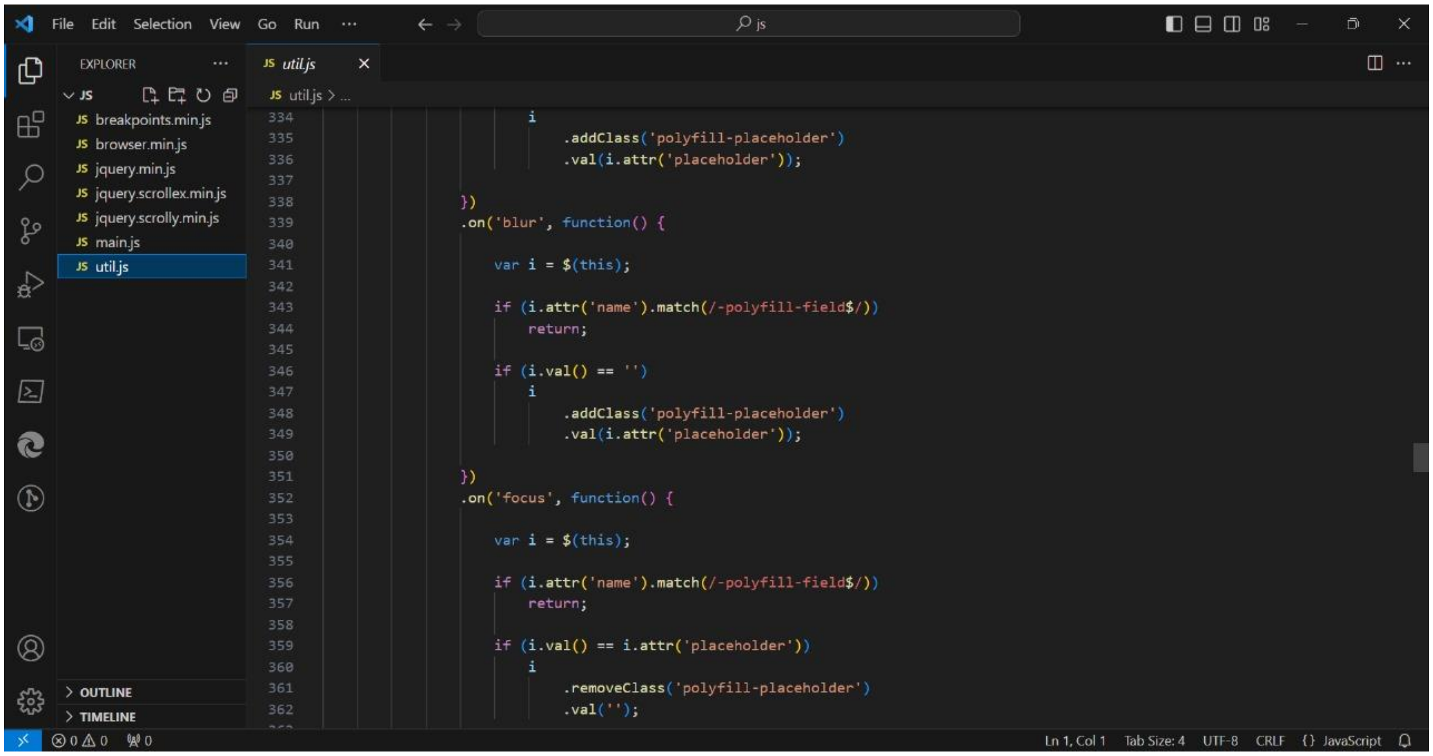
```
// Breakpoints.
    breakpoints({
        xlarge:   [ '1281px',  '1680px' ],
        large:    [ '1025px',  '1280px' ],
        medium:   [ '737px',   '1024px' ],
        small:    [ '481px',   '736px'  ],
        xsmall:   [ null,      '480px'  ],
    });

// Play initial animations on page load.
    $window.on('load', function() {
        window.setTimeout(function() {
            $body.removeClass('is-preload');
        }, 100);
    });

// Tweaks/fixes.

    // Polyfill: Object fit.
        if (!browser.canUse('object-fit')) {

            $('.image[data-position]').each(function() {

                var $this = $(this),
                    $img = $this.children('img');

                // Apply img as background.
                    $this
                        .css('background-image', 'url("' + $img.attr('src') + '")')
```



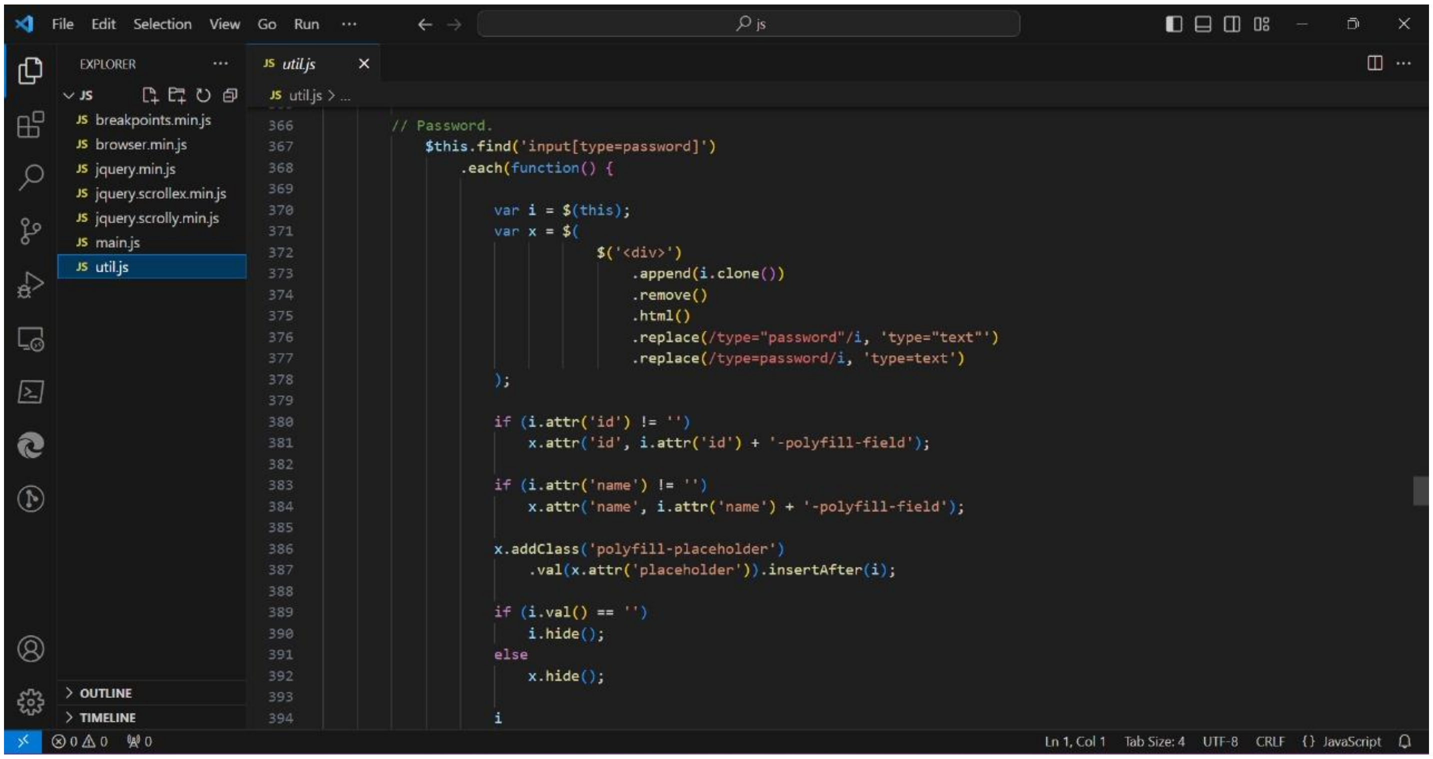
```
    // Nav.
        var $nav_a = $nav.find('a');

        $nav_a
            .addClass('scrolly')
            .on('click', function() {

                var $this = $(this);

                // External link? Bail.
                    if ($this.attr('href').charAt(0) != '#')
                        return;

                // Deactivate all links.
                    $nav_a.removeClass('active');

                // Activate link *and* lock it (so Scrollex doesn't try to activate other links as we're scrolling to this on
                    $this
                        .addClass('active')
                        .addClass('active-locked');

            })
            .each(function() {

                var $this = $(this),
                    id = $this.attr('href'),
                    $section = $(id);

                // No section for this link? Bail.
                    if ($section.length < 1)
```



```javascript
 * Apply "placeholder" attribute polyfill to one or more forms.
 * @return {jQuery} jQuery object.
 */
$.fn.placeholder = function() {

    // Browser natively supports placeholders? Bail.
        if (typeof (document.createElement('input')).placeholder != 'undefined')
            return $(this);

    // No elements?
        if (this.length == 0)
            return $this;

    // Multiple elements?
        if (this.length > 1) {

            for (var i=0; i < this.length; i++)
                $(this[i]).placeholder();

            return $this;

        }

    // Vars.
        var $this = $(this);

    // Text, TextArea.
        $this.find('input[type=text],textarea')
            .each(function() {
```

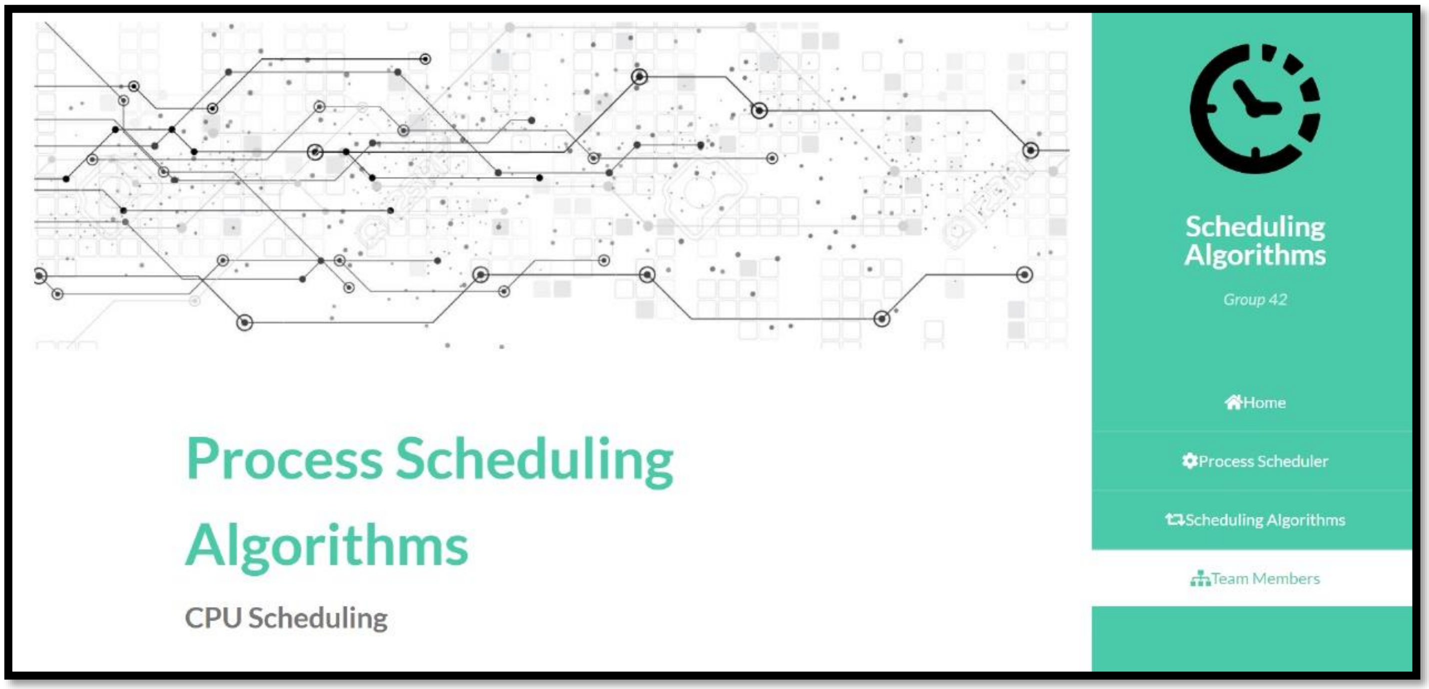


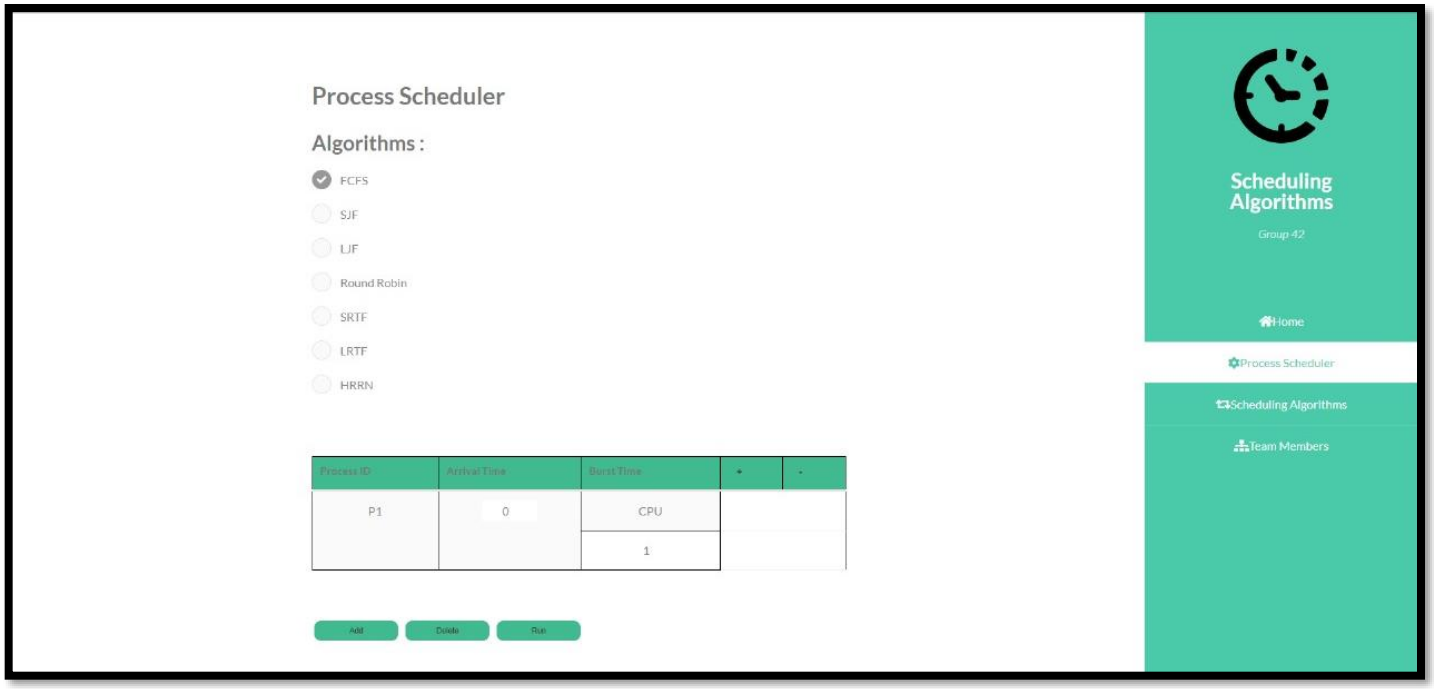
```javascript
                    i
                        .addClass('polyfill-placeholder')
                        .val(i.attr('placeholder'));

            })
            .on('blur', function() {

                var i = $(this);

                if (i.attr('name').match(/-polyfill-field$/))
                    return;

                if (i.val() == '')
                    i
                        .addClass('polyfill-placeholder')
                        .val(i.attr('placeholder'));

            })
            .on('focus', function() {

                var i = $(this);

                if (i.attr('name').match(/-polyfill-field$/))
                    return;

                if (i.val() == i.attr('placeholder'))
                    i
                        .removeClass('polyfill-placeholder')
                        .val('');
```



```
        // Password.
            $this.find('input[type=password]')
                .each(function() {

                    var i = $(this);
                    var x = $(
                                $('<div>')
                                    .append(i.clone())
                                    .remove()
                                    .html()
                                    .replace(/type="password"/i, 'type="text"')
                                    .replace(/type=password/i, 'type=text')
                    );

                    if (i.attr('id') != '')
                        x.attr('id', i.attr('id') + '-polyfill-field');

                    if (i.attr('name') != '')
                        x.attr('name', i.attr('name') + '-polyfill-field');

                    x.addClass('polyfill-placeholder')
                        .val(x.attr('placeholder')).insertAfter(i);

                    if (i.val() == '')
                        i.hide();
                    else
                        x.hide();

                    i
```

5. RESULT



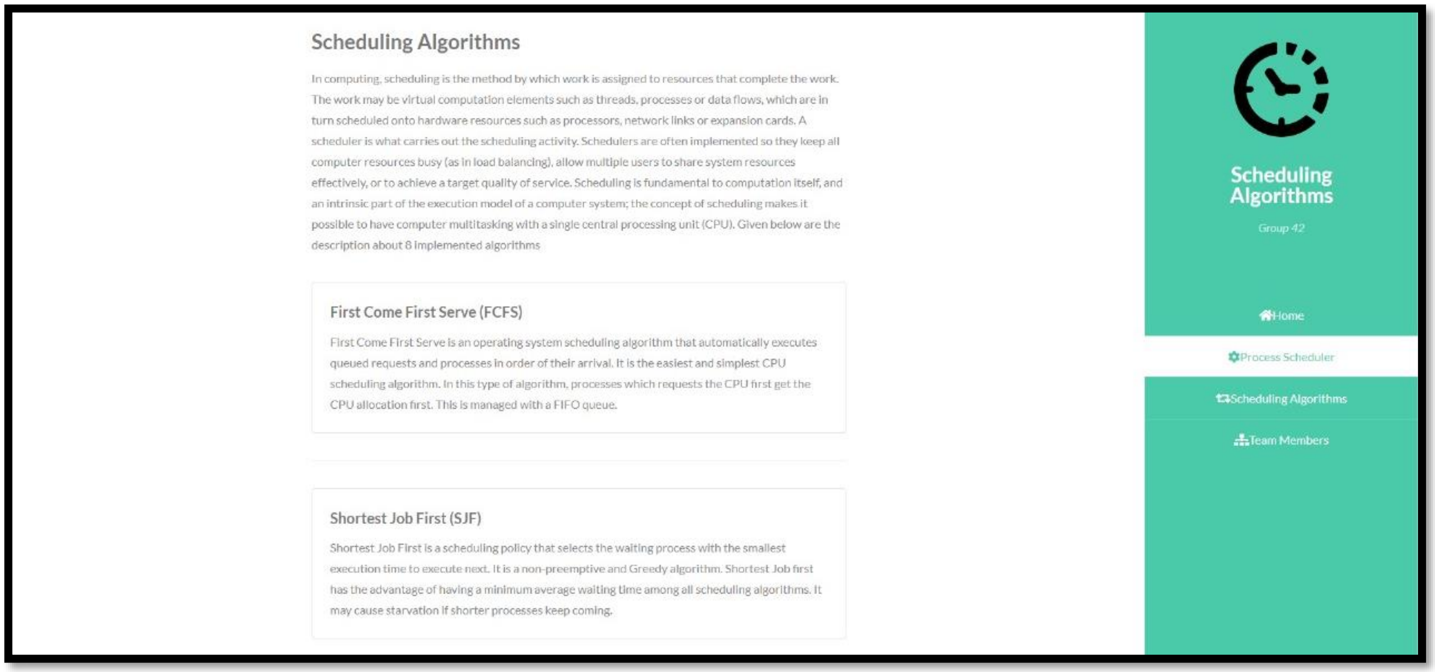


Process Scheduler

Algorithms :

[✓ FCFS]
SJF
LJF
Round Robin
SRTF
LRTF
HRRN

Process ID	Arrival Time	Burst Time	+	-
P1	0	CPU		
		1		



Scheduling Algorithms

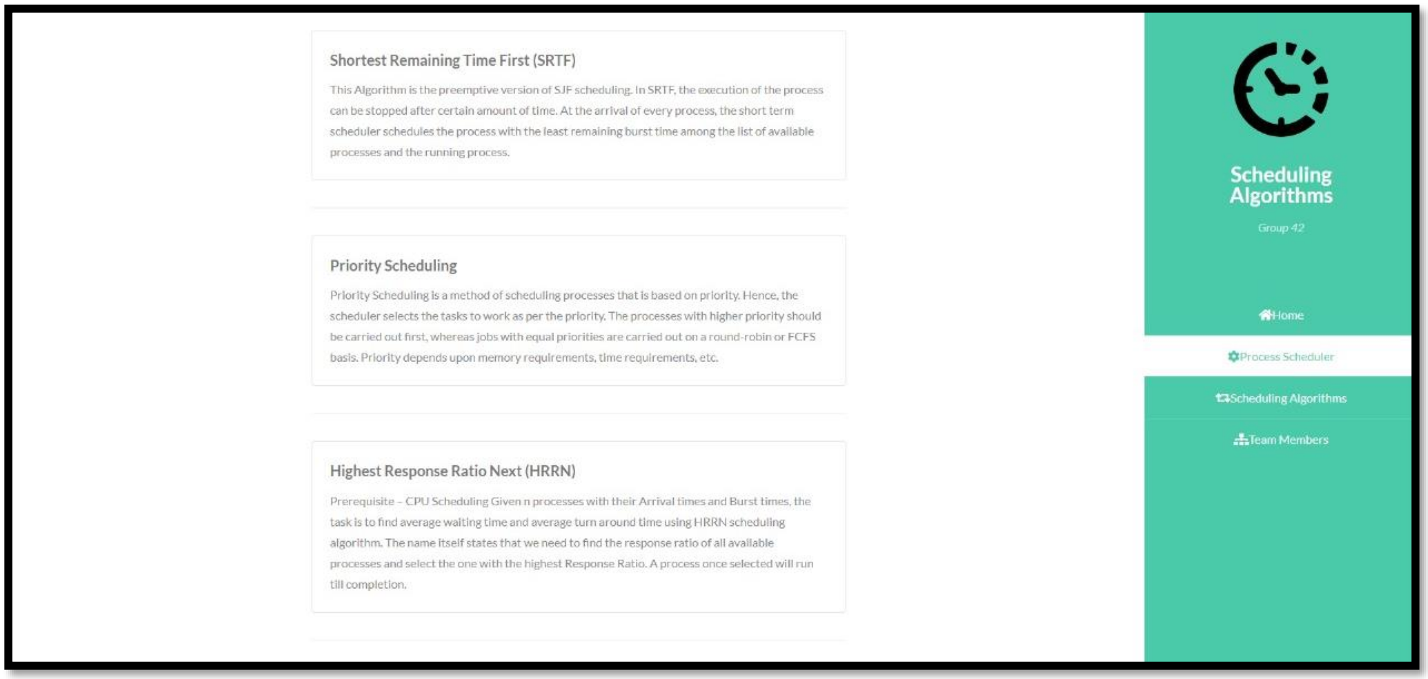
In computing, scheduling is the method by which work is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards. A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU). Given below are the description about 8 implemented algorithms

First Come First Serve (FCFS)

First Come First Serve is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first. This is managed with a FIFO queue.

Shortest Job First (SJF)

Shortest Job First is a scheduling policy that selects the waiting process with the smallest execution time to execute next. It is a non-preemptive and Greedy algorithm. Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms. It may cause starvation if shorter processes keep coming.



Shortest Remaining Time First (SRTF)

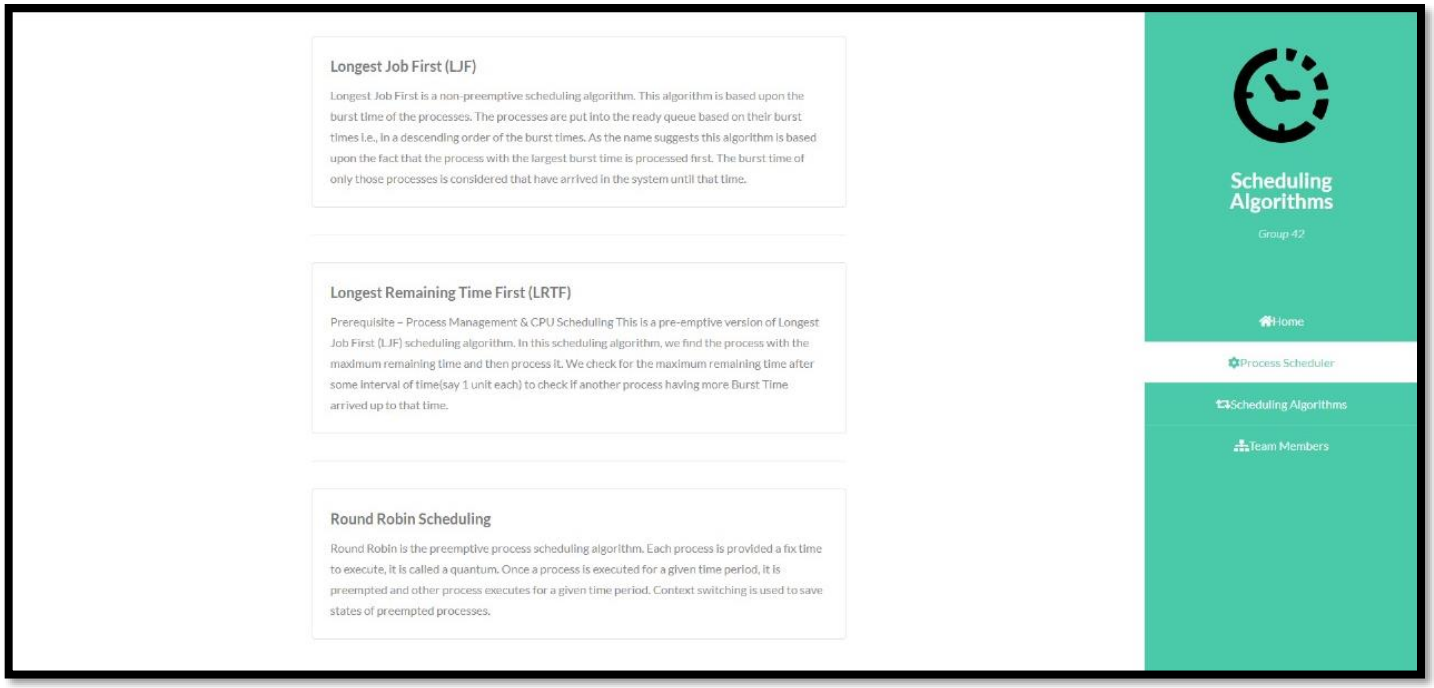
This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

Priority Scheduling

Priority Scheduling is a method of scheduling processes that is based on priority. Hence, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Highest Response Ratio Next (HRRN)

Prerequisite – CPU Scheduling Given n processes with their Arrival times and Burst times, the task is to find average waiting time and average turn around time using HRRN scheduling algorithm. The name itself states that we need to find the response ratio of all available processes and select the one with the highest Response Ratio. A process once selected will run till completion.



Longest Job First (LJF)

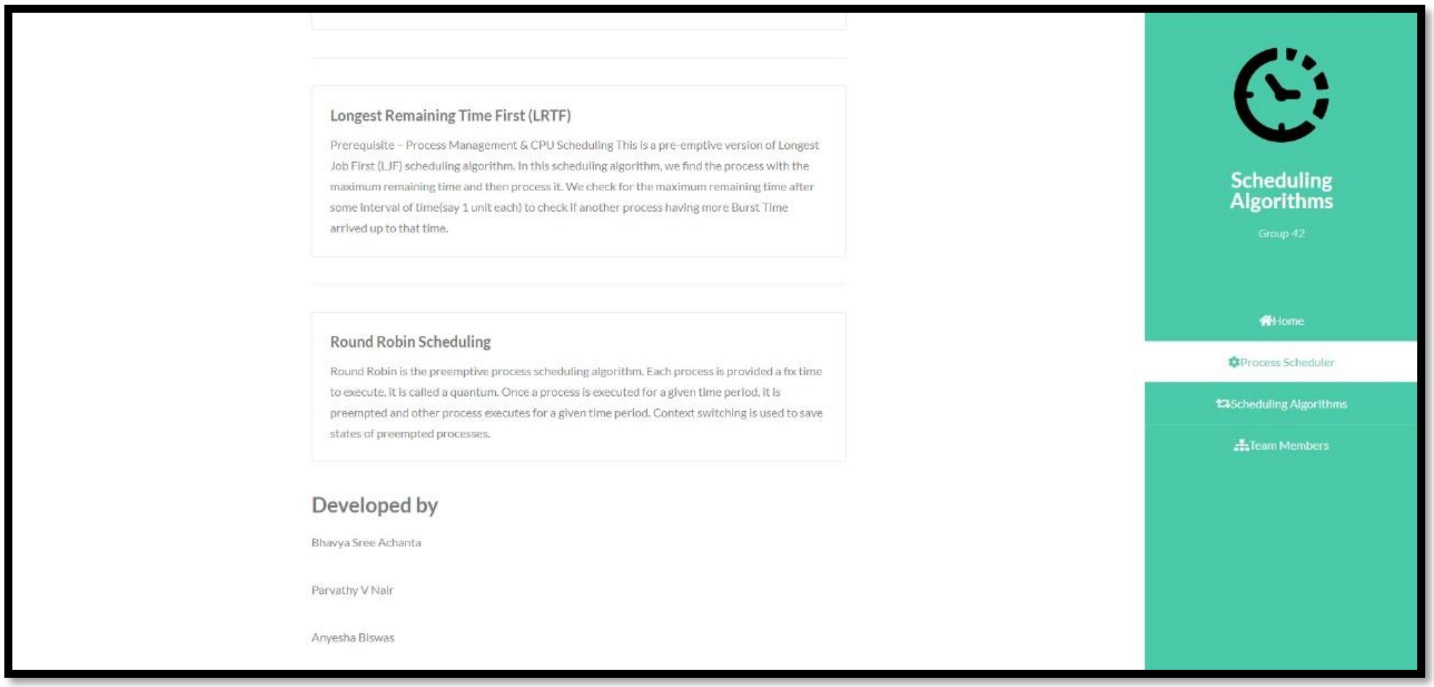
Longest Job First is a non-preemptive scheduling algorithm. This algorithm is based upon the burst time of the processes. The processes are put into the ready queue based on their burst times i.e., in a descending order of the burst times. As the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. The burst time of only those processes is considered that have arrived in the system until that time.

Longest Remaining Time First (LRTF)

Prerequisite – Process Management & CPU Scheduling This is a pre-emptive version of Longest Job First (LJF) scheduling algorithm. In this scheduling algorithm, we find the process with the maximum remaining time and then process it. We check for the maximum remaining time after some interval of time(say 1 unit each) to check if another process having more Burst Time arrived up to that time.

Round Robin Scheduling

Round Robin is the preemptive process scheduling algorithm. Each process is provided a fix time to execute, it is called a quantum. Once a process is executed for a given time period, it is preempted and other process executes for a given time period. Context switching is used to save states of preempted processes.



Longest Remaining Time First (LRTF)

Prerequisite – Process Management & CPU Scheduling This is a pre-emptive version of Longest Job First (LJF) scheduling algorithm. In this scheduling algorithm, we find the process with the maximum remaining time and then process it. We check for the maximum remaining time after some interval of time(say 1 unit each) to check if another process having more Burst Time arrived up to that time.

Round Robin Scheduling

Round Robin is the preemptive process scheduling algorithm. Each process is provided a fix time to execute, it is called a quantum. Once a process is executed for a given time period, it is preempted and other process executes for a given time period. Context switching is used to save states of preempted processes.

Developed by

Bhavya Sree Achanta

Parvathy V Nair

Anyesha Biswas

5. CONCLUSION

In summary, a process scheduling simulator is a critical tool for improving the efficiency and performance of multi-process systems. It addresses the challenges of unpredictable process behavior, enabling the development and optimization of scheduling algorithms.

Overall, it plays a pivotal role in enhancing real-world multi-process system efficiency.

In conclusion, a process management simulator in an operating system serves as a valuable tool for understanding, testing, and optimizing the orchestration of tasks. By mimicking real-world scenarios, it allows for the assessment of system performance, resource utilization, and the impact of various scheduling algorithms. This simulation framework aids in the development and refinement of robust process management strategies, contributing to the enhancement of overall operating system efficiency and reliability.

The choice of approach depends on the specific goals and requirements of the process management task. Process mining tools, machine learning algorithms, and domain-specific knowledge are often combined to develop effective detection systems for various aspects of process management.

6.