OS Project Documentation

Project: CPU Scheduling Simulator

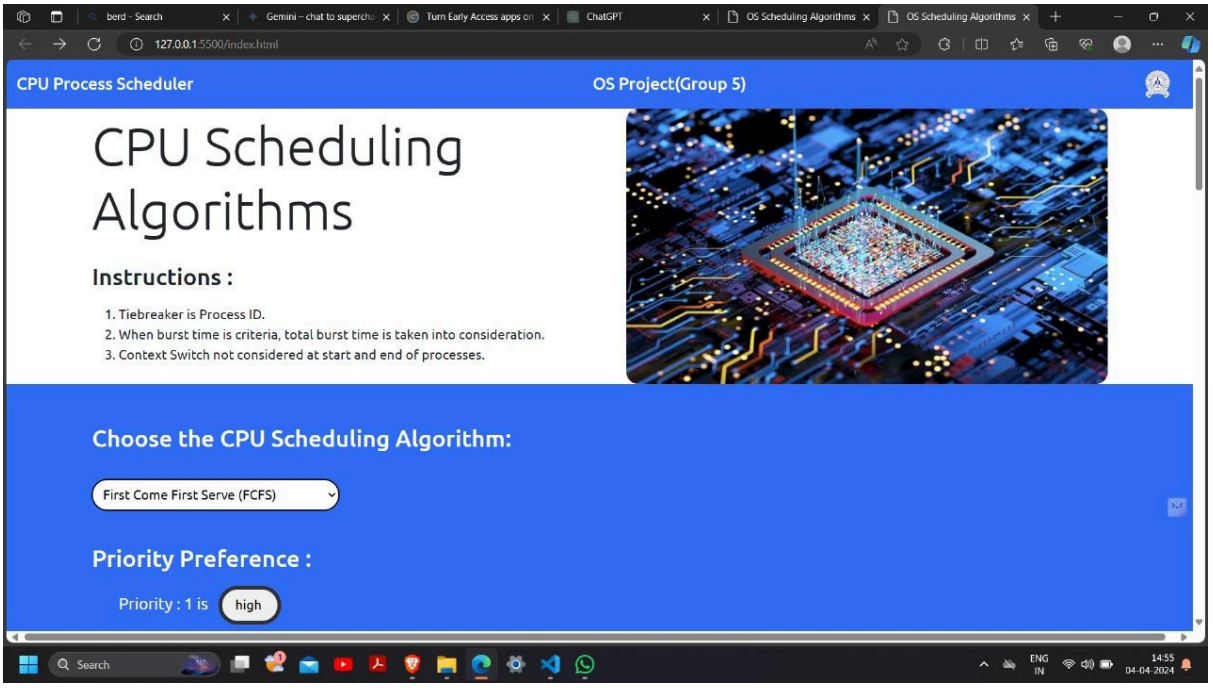
Members: 1.Ashish Singh(22JE0188)-Team Lead
2.Ashish Singh(22JE0187)
3.Ashok Mahala(22JE0190)
4.Aryan Meena(22JE0179)

Submission made to: Prof. Hari Om

Mentored by: Arshia Chaudhari Ma’am

Date of Submission:4th April 2024

Introduction:



This documentation outlines the design, features, and functionality of a web-based CPU process scheduling simulator. The simulator is built using HTML, CSS, and JavaScript and is designed to visualize various CPU scheduling algorithms. It takes user inputs such as algorithm type, arrival time, burst time, etc., and generates a Gantt chart, a final table, a timeline, and average values as outputs.

Features:

1.**Algorithm Selection**: Users can choose from various CPU scheduling algorithms, including First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), etc.
Input Parameters: Users can input the details of processes such as arrival time, burst time, priority (if applicable), and quantum time (for RR).
2.**Visualization**: The simulator provides visual representations of the scheduling process, including Gantt chart and timeline.

Output Analysis: The simulator generates a final table displaying the order of execution, waiting time, turnaround time, etc.

3.**Performance Metrics:** Average waiting time, turnaround time, and other performance metrics are calculated and displayed.

Interactive Interface: The interface is user-friendly and interactive, allowing users to easily input parameters and view results.

Design and Implementation:

1. **Frontend (HTML/CSS)**:

index.html: This file contains the structure of the web application, including input forms, buttons, and output display areas.

style.css: Defines the styles and layout of the HTML elements for a visually appealing interface.

2. **Backend (JavaScript):**

script.js: Handles the logic of the CPU scheduling simulator.

Algorithms.js: Contains implementations of various CPU scheduling algorithms such as FCFS, SJF, RR, etc.

3. **Input Handling:**

User inputs such as algorithm selection, process details, etc., are captured using HTML forms and JavaScript event listeners. Input validation ensures that the user provides valid data before simulation.

4. **Simulation Logic:**

Based on the selected algorithm, the simulator uses appropriate logic to schedule processes.

Algorithms.js contains functions for each scheduling algorithm to calculate waiting time, turnaround time, etc.

5. **Visualization:**

Gantt Chart: Generated dynamically using HTML/CSS based on the scheduled processes.

Timeline: A visual representation of the scheduling timeline, highlighting when each process starts and ends.

6. **Output Generation:**

A final table is generated displaying the order of execution, waiting time, turnaround time, etc.

Performance metrics such as average waiting time, turnaround time, etc., are calculated and displayed.

Usage:

- Open the web application in a web browser.
- Select the desired CPU scheduling algorithm.
- Input details of processes including arrival time, burst time, etc.
- Click on the "Calculate" button to run the simulation.
- View the generated Gantt chart, final table, timeline, and performance metrics.

Conclusion:

This web-based CPU process scheduling simulator provides a user-friendly interface for visualizing and analyzing various CPU scheduling algorithms. It enables users to understand the behaviour and performance of different scheduling policies and can be a valuable tool for students and professionals studying or working in the field of operating systems.

Workflow:

1.Initialization:

User opens the web application in a web browser.
Interface Display:

The user is presented with an interface containing input forms, algorithm selection dropdown, and output display areas.
HTML/CSS renders the layout and styles the interface elements.
2.Input Gathering:

User selects a CPU scheduling algorithm from the dropdown menu.
User inputs details of processes including arrival time, burst time, priority (if applicable), and quantum time (for RR).
JavaScript event listeners capture user inputs and store them in variables.
3.Input Validation:

JavaScript validates user inputs to ensure they are within acceptable ranges and formats.
Error messages are displayed if input is invalid, prompting the user to correct it.
4.Simulation Initialization:

Upon successful validation, the simulator initializes by retrieving user inputs.
5.Scheduling Algorithm Execution:

Based on the selected algorithm, the simulator executes the corresponding scheduling logic.

Functions from Algorithms.js are called to perform scheduling calculations.
6.Gantt Chart Generation:

Using the calculated scheduling data, the simulator dynamically generates a Gantt chart visualizing the process execution timeline.
HTML/CSS is updated to display the Gantt chart.
7.Final Table Generation:

A final table is generated displaying the order of execution, waiting time, turnaround time, etc.
JavaScript updates the HTML to display the final table.
8.Timeline Visualization:

The simulator generates a timeline visual representation of the scheduling process, highlighting when each process starts and ends.
HTML/CSS is updated to display the timeline.
9.Performance Metrics Calculation:

Performance metrics such as average waiting time, average turnaround time, etc., are calculated based on the scheduling data.
JavaScript computes these metrics and updates the HTML to display them.
10.Output Display:

The generated Gantt chart, final table, timeline, and performance metrics are displayed to the user.
11.Interaction:

The user can interact with the displayed outputs, zooming in/out on the Gantt chart, hovering over elements for details, etc.
12.Simulation Replay/Modification:

Optionally, the user can modify input parameters and rerun the simulation to observe the effects of different settings.
13.Conclusion:

The user gains insights into the behaviour and performance of the selected CPU scheduling algorithm through visualizations and analysis provided by the simulator.
14.End:

The user can close the web application or perform further analyses as needed.

Tech Stack:

Frontend:

HTML (HyperText Markup Language):

Used for creating the structure and content of the web pages.
Defines the layout of input forms, buttons, output display areas, etc.
CSS (Cascading Style Sheets):

Styles the HTML elements to improve the visual presentation of the web application.
Defines the colors, fonts, sizes, and layouts of the user interface.
JavaScript:

Provides interactivity and dynamic behaviour to the web application.
Captures user inputs, performs input validation, and updates the interface accordingly.
Dynamically generates visual elements such as Gantt chart, final table, and timeline.
Backend:

JavaScript:
Handles the logic and functionality of the CPU process scheduling simulator.
Manages the simulation process, scheduling algorithms execution, and output generation.
Algorithm Implementation:

Algorithms.js (JavaScript):
Contains implementations of various CPU scheduling algorithms such as FCFS, SJF, RR, etc.
Provides functions to calculate waiting time, turnaround time, and other scheduling parameters based on the selected algorithm.
Data Management:

Variables (JavaScript):
Stores user inputs such as algorithm selection, process details, etc.
Manages scheduling data during the simulation process.
Visualization:

HTML/CSS (Gantt Chart):

Dynamically generates a Gantt chart visualizing the process execution timeline.
Utilizes CSS for styling and layout adjustments.
HTML/CSS (Timeline):

Generates a timeline visual representation of the scheduling process, highlighting when each process starts and ends.
Utilizes CSS for styling and layout adjustments.
Input Validation:

JavaScript:
Validates user inputs to ensure they are within acceptable ranges and formats.
Displays error messages if input is invalid, prompting the user to correct it.
Output Generation:

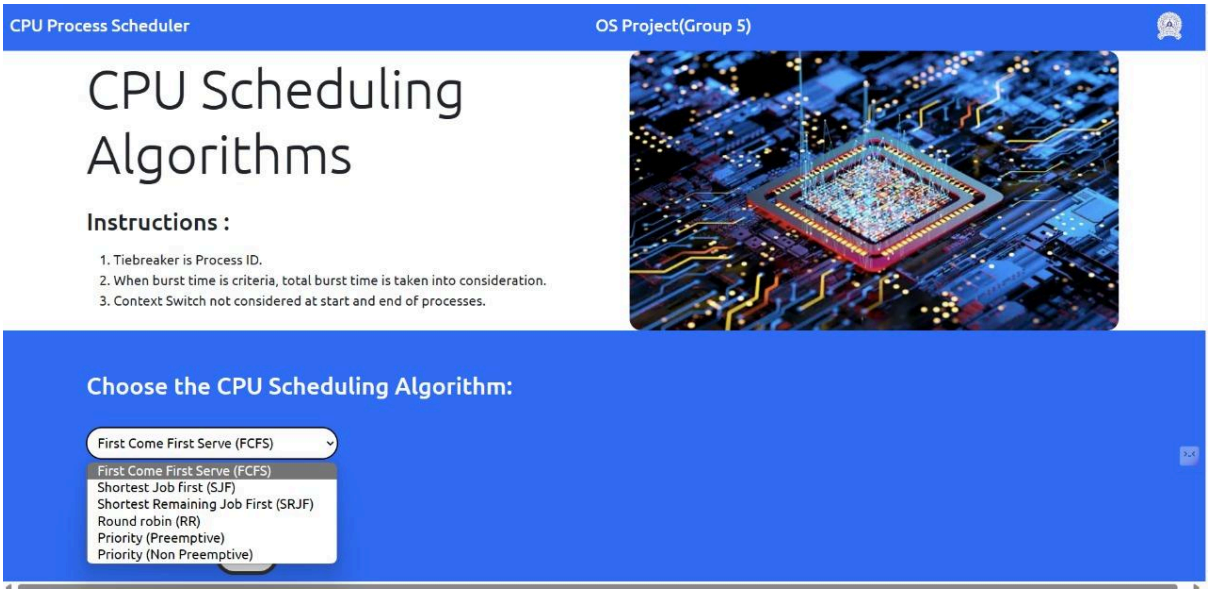
HTML/CSS (Final Table):
Generates a final table displaying the order of execution, waiting time, turnaround time, etc.
Utilizes CSS for styling and layout adjustments.
Performance Metrics Calculation:

JavaScript:
Computes performance metrics such as average waiting time, average turnaround time, etc.
Updates the HTML to display the calculated metrics.
Interactivity:

JavaScript:
Enables user interaction with the displayed outputs, such as zooming in/out on the Gantt chart, hovering over elements for details, etc.

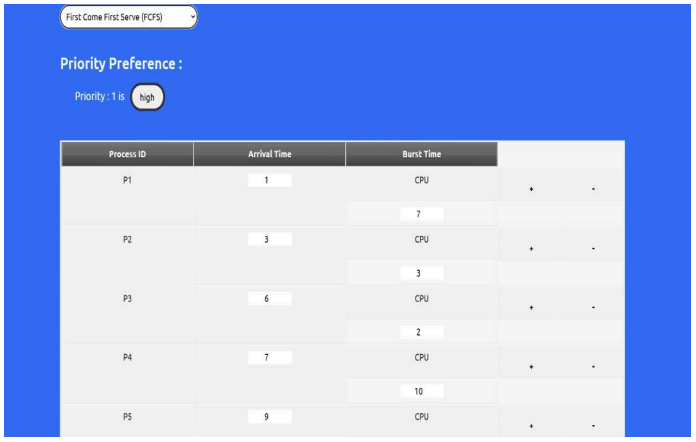
Conclusion:
The tech stack used for the CPU process scheduling simulator includes HTML, CSS, and JavaScript for frontend and backend development. JavaScript is primarily responsible for handling the logic, interactivity, and data management of the simulator, while HTML and CSS are utilized for creating the user interface and visualizations. Additionally, algorithm implementations are provided in JavaScript for executing various CPU scheduling algorithms, and input validation ensures the integrity of user inputs.

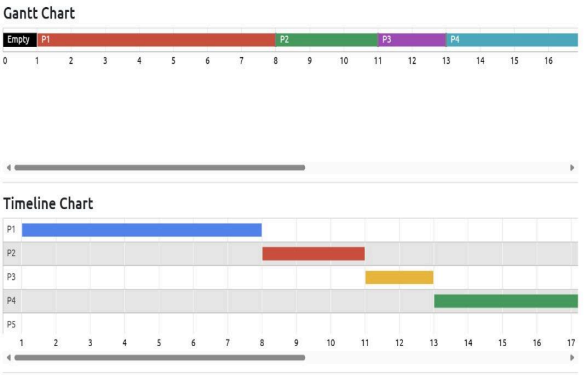
Insights of the Web Application:



Insight of the HomePage

FCFS(First Come First Serve) Working:





Final Table

Process	Arrival Time	Total Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	1	7	8	7	0	0
P2	3	3	11	8	5	5
P3	6	2	13	7	5	5
P4	7	10	23	16	6	6
P5	9	9	32	23	14	14

CPU Utilization : 96.875%

Throughput : 0.15625

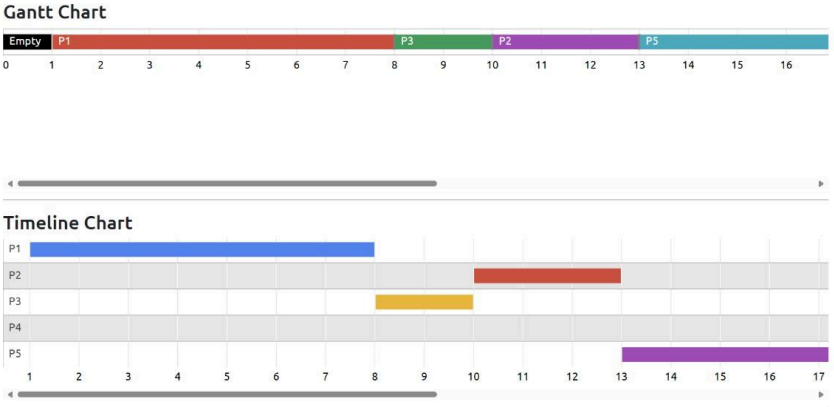
Average Completion Time : 17.4

Average Turn Around Time : 12.2

Average Waiting Time : 6

Average Response Time : 6

Shortest Job First(SJF)



Final Table

Process	Arrival Time	Total Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	1	7	8	7	0	0
P2	3	3	13	10	7	7
P3	6	2	10	4	2	2
P4	7	10	32	25	15	15
P5	9	9	22	13	4	4

CPU Utilization : 96.875%

Throughput : 0.15625

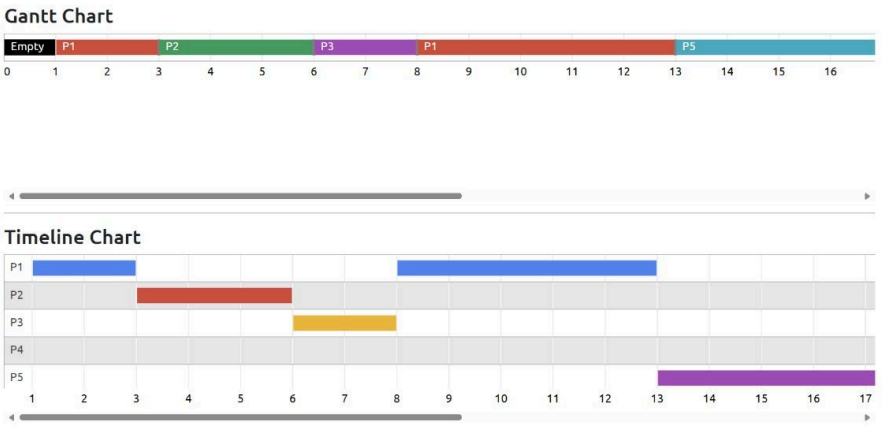
Average Completion Time : 17

Average Turn Around Time : 11.8

Average Waiting Time : 5.6

Average Response Time : 5.6

Shortest Remaining Job First(SRJF)



Final Table

Process	Arrival Time	Total Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	1	7	13	12	5	0
P2	3	3	6	3	0	0
P3	6	2	8	2	0	0
P4	7	10	32	25	15	15
P5	9	9	22	13	4	4

CPU Utilization : 96.875%

Throughput : 0.15625

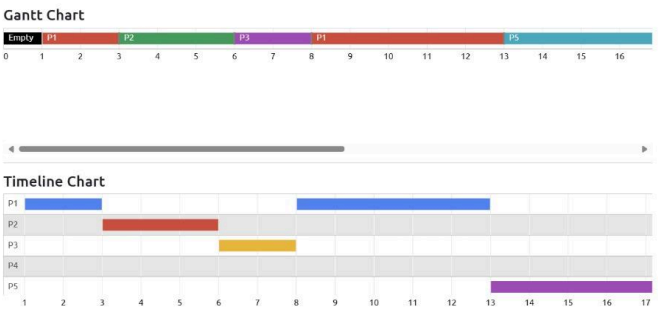
Average Completion Time : 16.2

Average Turn Around Time : 11

Average Waiting Time : 4.8

Average Response Time : 3.8

Round Robin:



Final Table

Process	Arrival Time	Total Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	1	7	13	12	5	0
P2	3	3	6	3	0	0
P3	6	2	8	2	0	0
P4	7	10	32	25	15	15
P5	9	9	22	13	4	4

CPU Utilization : 96.875%

Throughput : 0.15625

Average Completion Time : 16.2

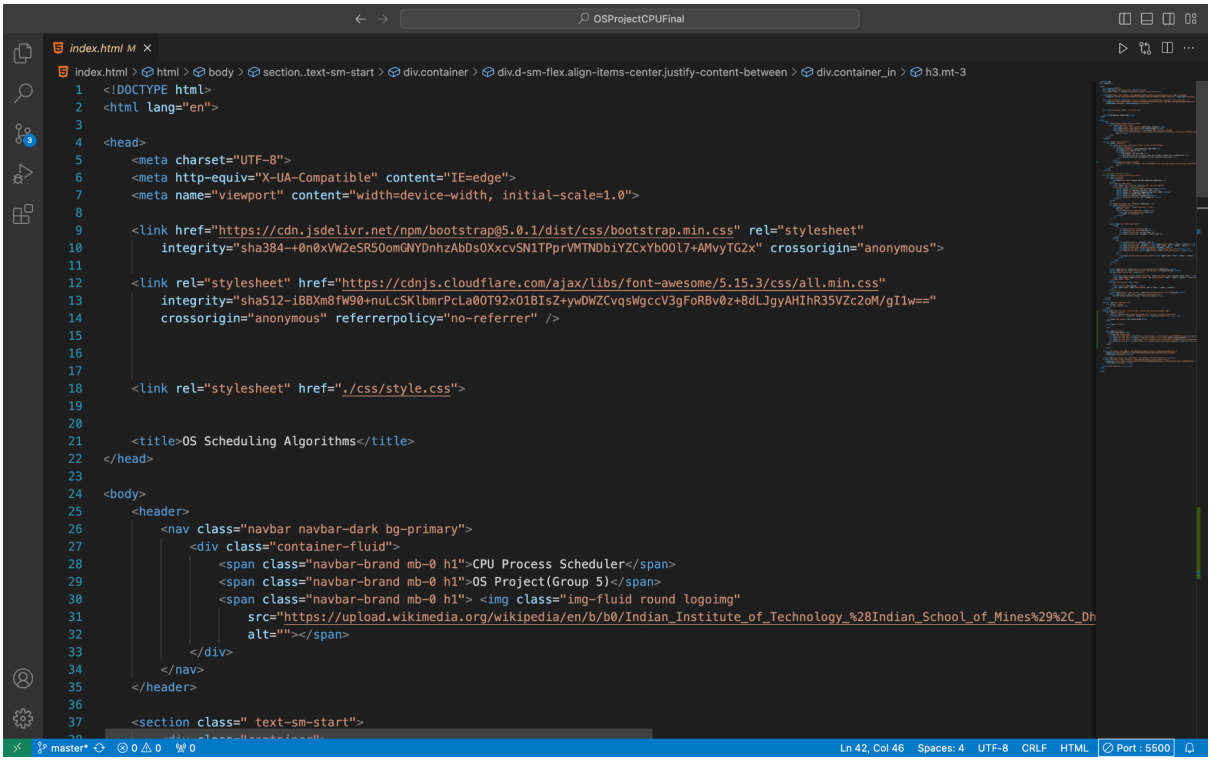
Average Turn Around Time : 11

Average Waiting Time : 4.8

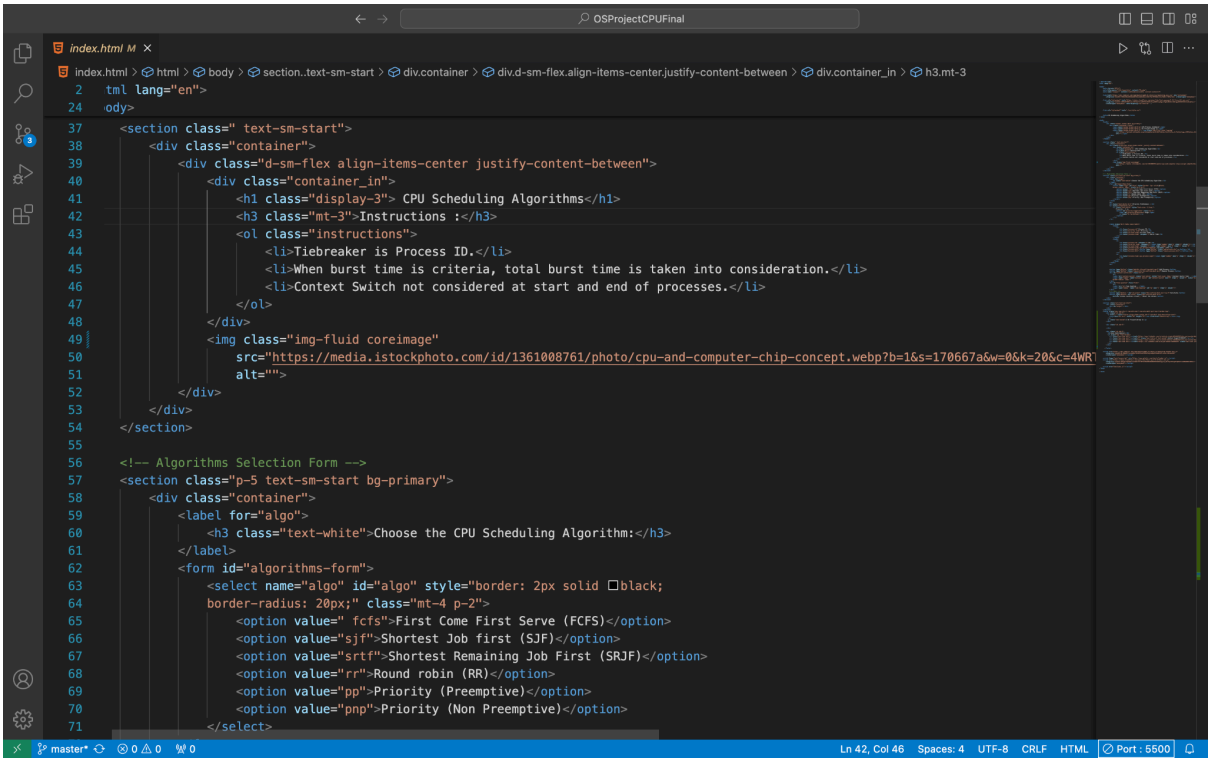
Average Response Time : 3.8

Turn over to next page for code snippets

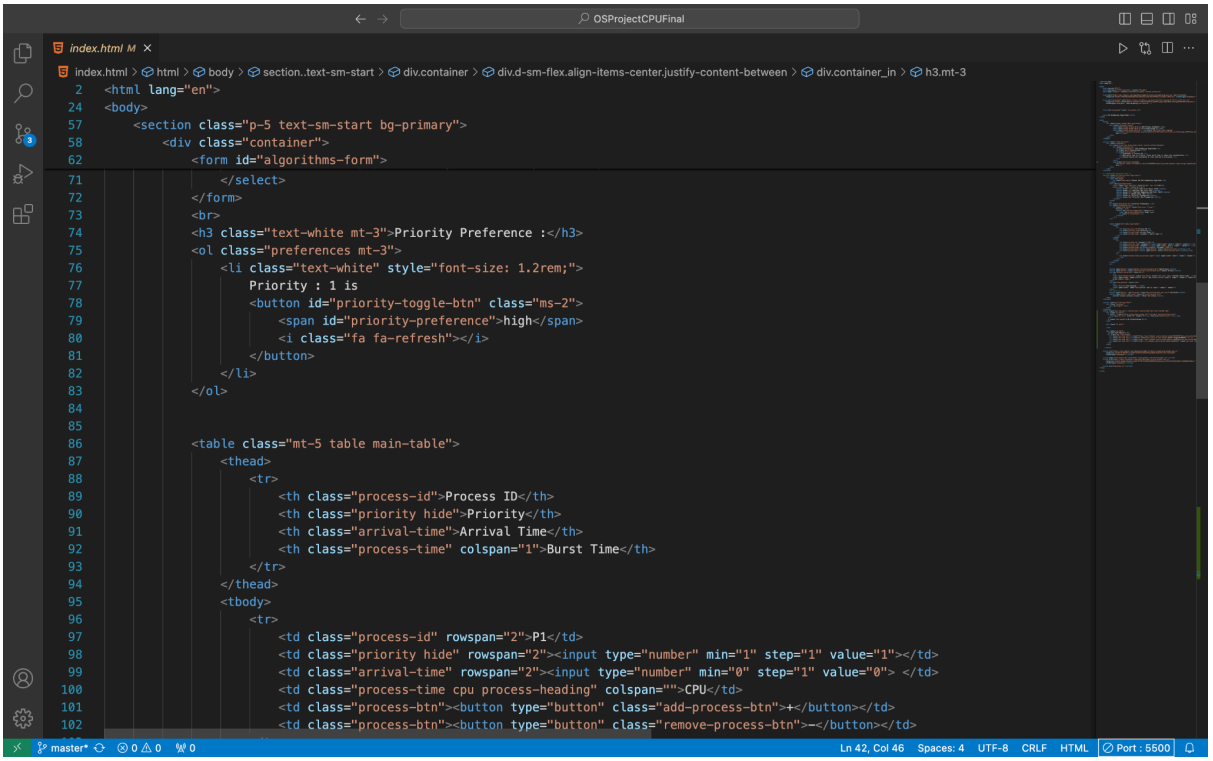
Code Snippets:



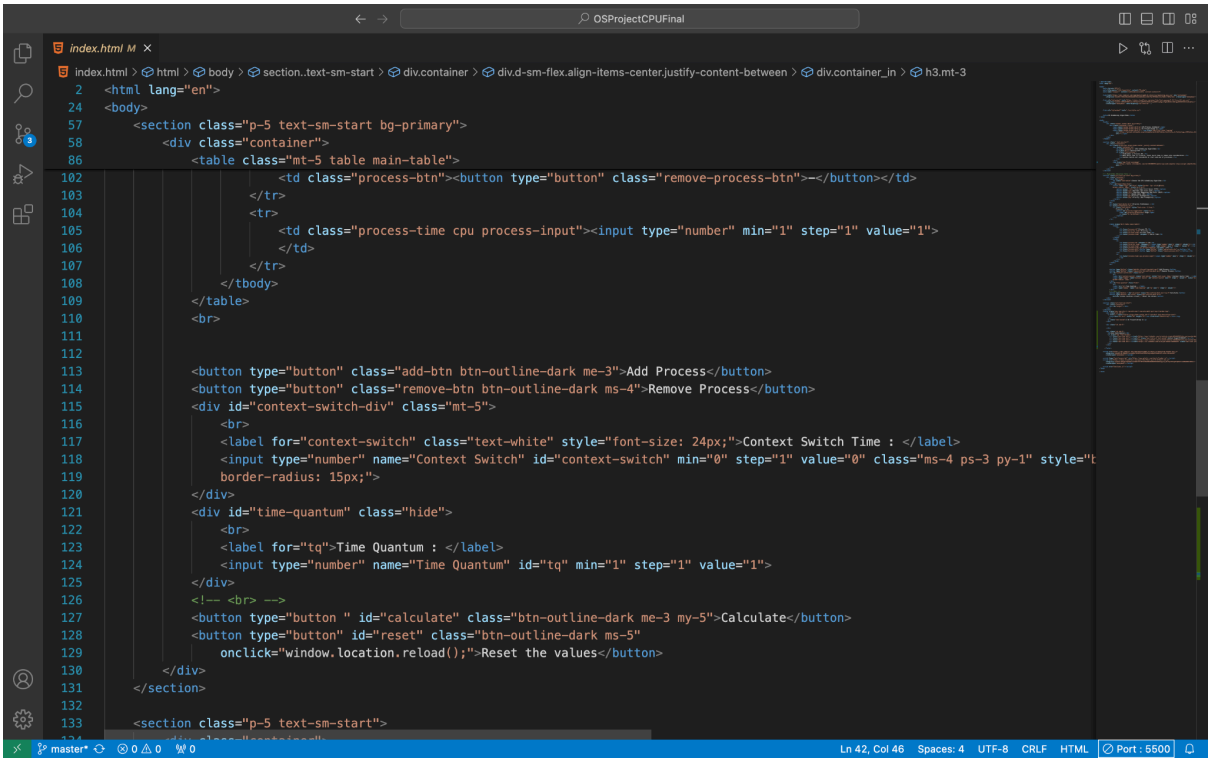
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
        integrity="sha512-iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM/gI1w=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />


    <link rel="stylesheet" href="./css/style.css">


    <title>OS Scheduling Algorithms</title>
</head>

<body>
    <header>
        <nav class="navbar navbar-dark bg-primary">
            <div class="container-fluid">
                <span class="navbar-brand mb-0 h1">CPU Process Scheduler</span>
                <span class="navbar-brand mb-0 h1">OS Project(Group 5)</span>
                <span class="navbar-brand mb-0 h1"> <img class="img-fluid round logoimg"
                    src="https://upload.wikimedia.org/wikipedia/en/b/b0/Indian_Institute_of_Technology_%28Indian_School_of_Mines%29%2C_Dh
                    alt=""></span>
            </div>
        </nav>
    </header>

    <section class=" text-sm-start">
```



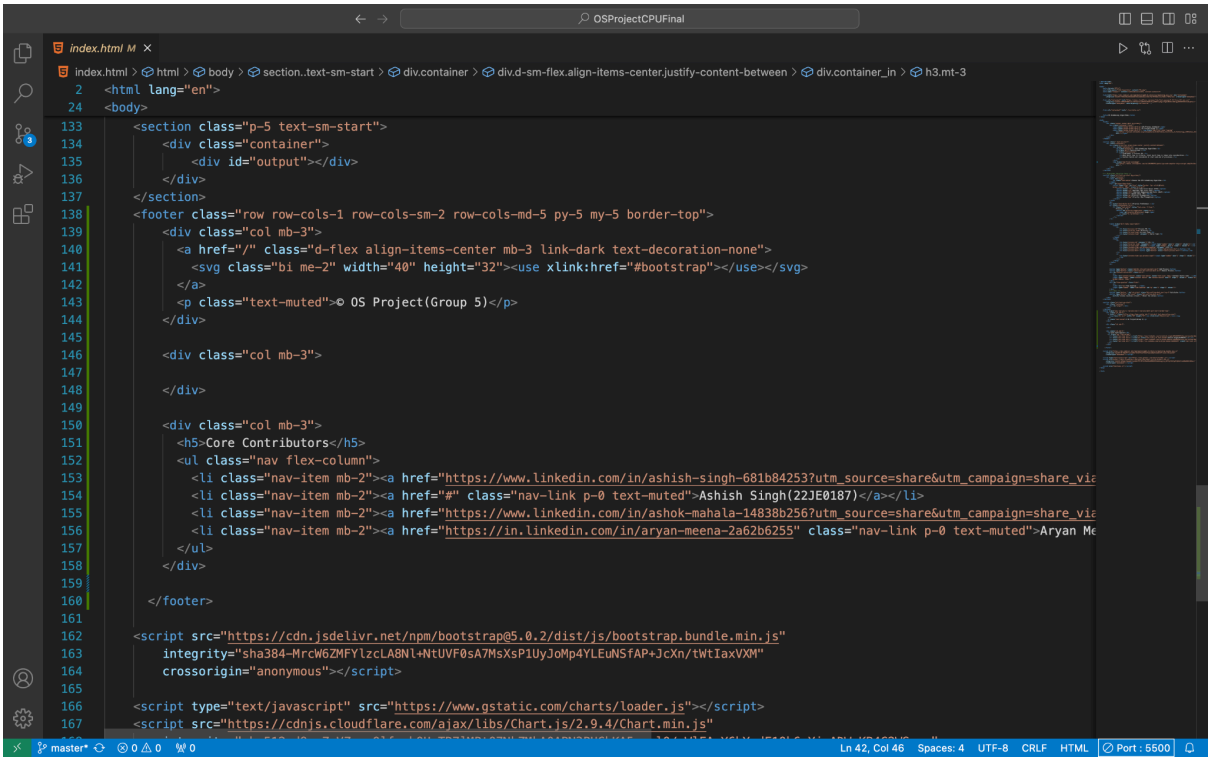
```
    tml lang="en">
    ody>
    <section class=" text-sm-start">
        <div class="container">
            <div class="d-sm-flex align-items-center justify-content-between">
                <div class="container_in">
                    <h1 class="display-3"> CPU Scheduling Algorithms</h1>
                    <h3 class="mt-3">Instructions :</h3>
                    <ol class="instructions">
                        <li>Tiebreaker is Process ID.</li>
                        <li>When burst time is criteria, total burst time is taken into consideration.</li>
                        <li>Context Switch not considered at start and end of processes.</li>
                    </ol>
                </div>
                <img class="img-fluid coreimage"
                    src="https://media.istockphoto.com/id/1361008761/photo/cpu-and-computer-chip-concept.webp?b=1&s=170667a&w=0&k=20&c=4WR
                    alt="">
            </div>
        </div>
    </section>

    <!-- Algorithms Selection Form -->
    <section class="p-5 text-sm-start bg-primary">
        <div class="container">
            <label for="algo">
                <h3 class="text-white">Choose the CPU Scheduling Algorithm:</h3>
            </label>
            <form id="algorithms-form">
                <select name="algo" id="algo" style="border: 2px solid black;
                border-radius: 20px;" class="mt-4 p-2">
                    <option value=" fcfs">First Come First Serve (FCFS)</option>
                    <option value="sjf">Shortest Job first (SJF)</option>
                    <option value="srtf">Shortest Remaining Job First (SRJF)</option>
                    <option value="rr">Round robin (RR)</option>
                    <option value="pp">Priority (Preemptive)</option>
                    <option value="pnp">Priority (Non Preemptive)</option>
                </select>
```

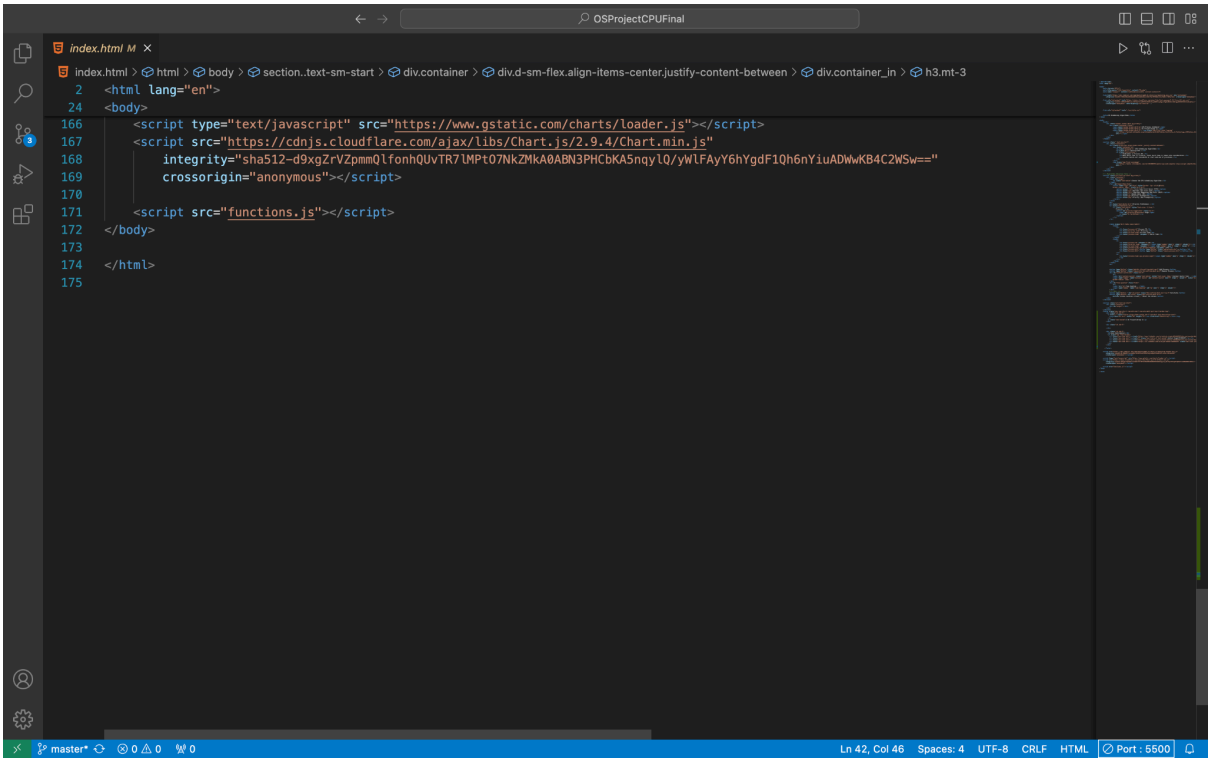


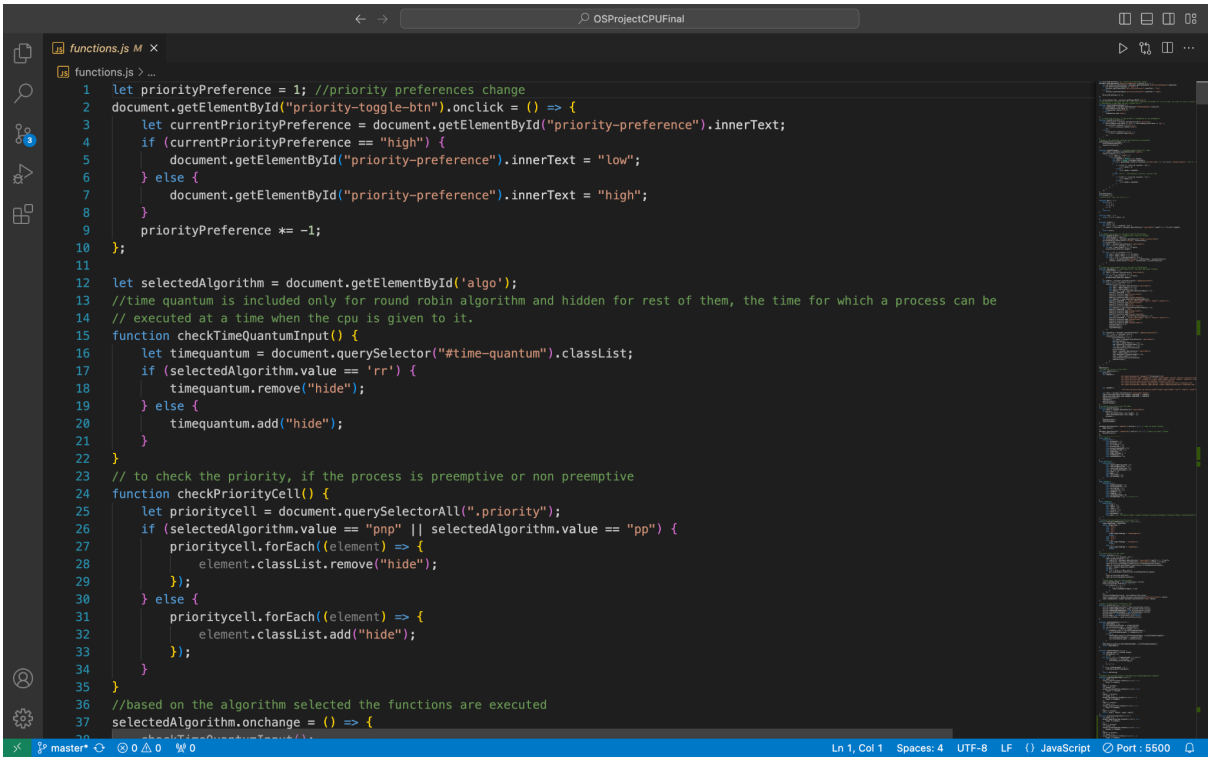

```
  <html lang="en">
  <body>
      <section class="p-5 text-sm-start bg-primary">
          <div class="container">
              <form id="algorithms-form">
                  </select>
              </form>
              <br>
              <h3 class="text-white mt-3">Priority Preference :</h3>
              <ol class="preferences mt-3">
                  <li class="text-white" style="font-size: 1.2rem;">
                      Priority : 1 is
                      <button id="priority-toggle-btn" class="ms-2">
                          <span id="priority-preference">high</span>
                          <i class="fa fa-refresh"></i>
                      </button>
                  </li>
              </ol>


              <table class="mt-5 table main-table">
                  <thead>
                      <tr>
                          <th class="process-id">Process ID</th>
                          <th class="priority hide">Priority</th>
                          <th class="arrival-time">Arrival Time</th>
                          <th class="process-time" colspan="1">Burst Time</th>
                      </tr>
                  </thead>
                  <tbody>
                      <tr>
                          <td class="process-id" rowspan="2">P1</td>
                          <td class="priority hide" rowspan="2"><input type="number" min="1" step="1" value="1"></td>
                          <td class="arrival-time" rowspan="2"><input type="number" min="0" step="1" value="0"> </td>
                          <td class="process-time cpu process-heading" colspan="">CPU</td>
                          <td class="process-btn"><button type="button" class="add-process-btn">+</button></td>
                          <td class="process-btn"><button type="button" class="remove-process-btn">-</button></td>
```

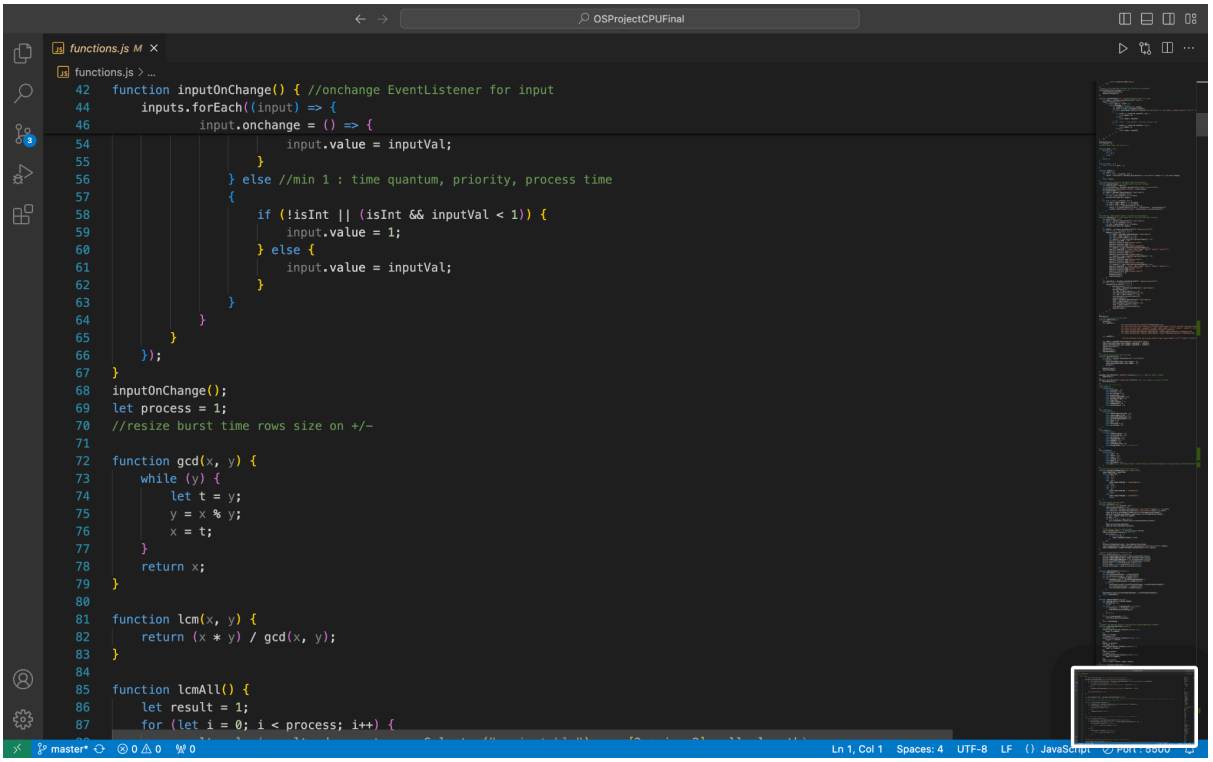


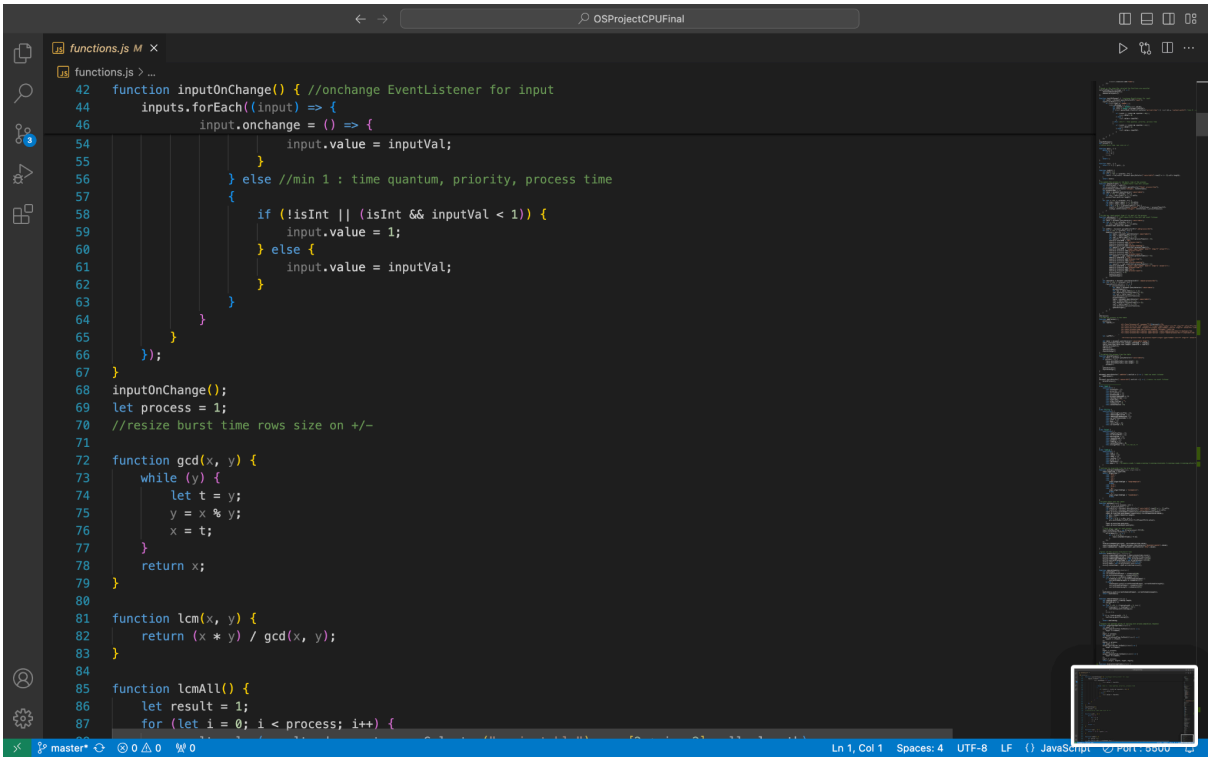

```
  <html lang="en">
  <body>
      <section class="p-5 text-sm-start bg-primary">
          <div class="container">
              <table class="mt-5 table main-table">
                          <td class="process-btn"><button type="button" class="remove-process-btn">-</button></td>
                      </tr>
                      <tr>
                          <td class="process-time cpu process-input"><input type="number" min="1" step="1" value="1">
                          </td>
                      </tr>
                  </tbody>
              </table>
              <br>


              <button type="button" class="add-btn btn-outline-dark me-3">Add Process</button>
              <button type="button" class="remove-btn btn-outline-dark ms-4">Remove Process</button>
              <div id="context-switch-div" class="mt-5">
                  <br>
                  <label for="context-switch" class="text-white" style="font-size: 24px;">Context Switch Time : </label>
                  <input type="number" name="Context Switch" id="context-switch" min="0" step="1" value="0" class="ms-4 ps-3 py-1" style="b
                  border-radius: 15px;">
              </div>
              <div id="time-quantum" class="hide">
                  <br>
                  <label for="tq">Time Quantum : </label>
                  <input type="number" name="Time Quantum" id="tq" min="1" step="1" value="1">
              </div>
              <!-- <br> -->
              <button type="button " id="calculate" class="btn-outline-dark me-3 my-5">Calculate</button>
              <button type="button" id="reset" class="btn-outline-dark ms-5"
                  onclick="window.location.reload();">Reset the values</button>
          </div>
      </section>

      <section class="p-5 text-sm-start">
```

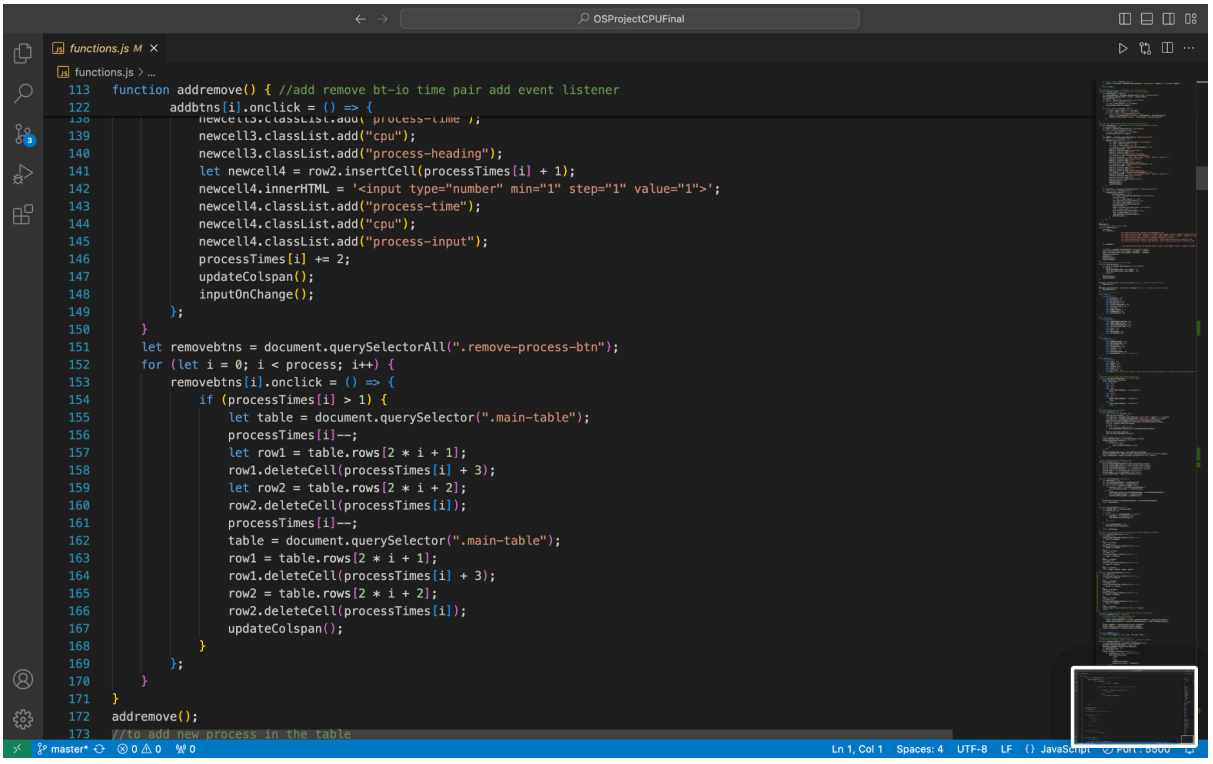

OSProjectCPUFinal
index.html M
index.html > html > body > section..text-sm-start > div.container > div.d-sm-flex.align-items-center.justify-content-between > div.container_in > h3.mt-3
2 <html lang="en">
24 <body>
133 <section class="p-5 text-sm-start">
134
135
136
137 </section>
138 <footer class="row row-cols-1 row-cols-sm-2 row-cols-md-5 py-5 my-5 border-top">
139
140
141 <svg class="bi me-2" width="40" height="32"><use xlink:href="#bootstrap"></use></svg>
142
143 © OS Project(Group 5)
144
145
146
147
148
149
150
151 <h5>Core Contributors</h5>
152 <ul class="nav flex-column">
153 <li class="nav-item mb-2"><a href="https://www.linkedin.com/in/ashish-singh-681b84253?utm_source=share&utm_campaign=share_via
154 <li class="nav-item mb-2">Ashish Singh(22JE0187)
155 <li class="nav-item mb-2"><a href="https://www.linkedin.com/in/ashok-mahala-14838b256?utm_source=share&utm_campaign=share_via
156 <li class="nav-item mb-2"><a href="https://in.linkedin.com/in/aryan-meena-2a62b6255" class="nav-link p-0 text-muted">Aryan Me
157
158
159
160 </footer>
161
162 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
163 integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
164 crossorigin="anonymous"></script>
165
166 <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
167 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"
master*
0 0
0
Ln 42, Col 46
Spaces: 4
UTF-8
CRLF
HTML
Port : 5500

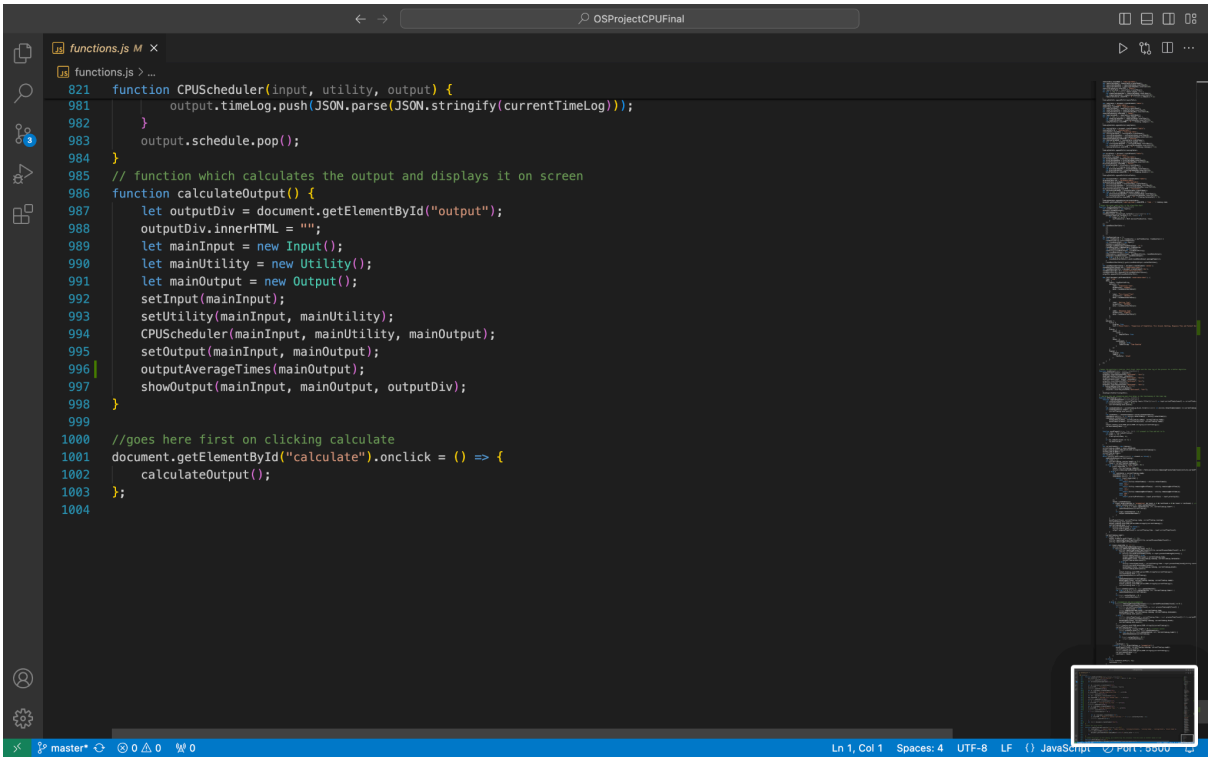

OSProjectCPUFinal
index.html M
index.html > html > body > section..text-sm-start > div.container > div.d-sm-flex.align-items-center.justify-content-between > div.container_in > h3.mt-3
2 <html lang="en">
24 <body>
166 <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
167 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"
168 integrity="sha512-d9xgZrVZpmmQlfonhQUvTR7lMPtO7NkZMkA0ABN3PHCbKA5nqylQ/yWlFAyY6hYgdF1Qh6nYiuADWwKB4C2WSw=="
169 crossorigin="anonymous"></script>
170
171 <script src="functions.js"></script>
172 </body>
173
174 </html>
175
master*
0 0
0
Ln 42, Col 46
Spaces: 4
UTF-8
CRLF
HTML
Port : 5500

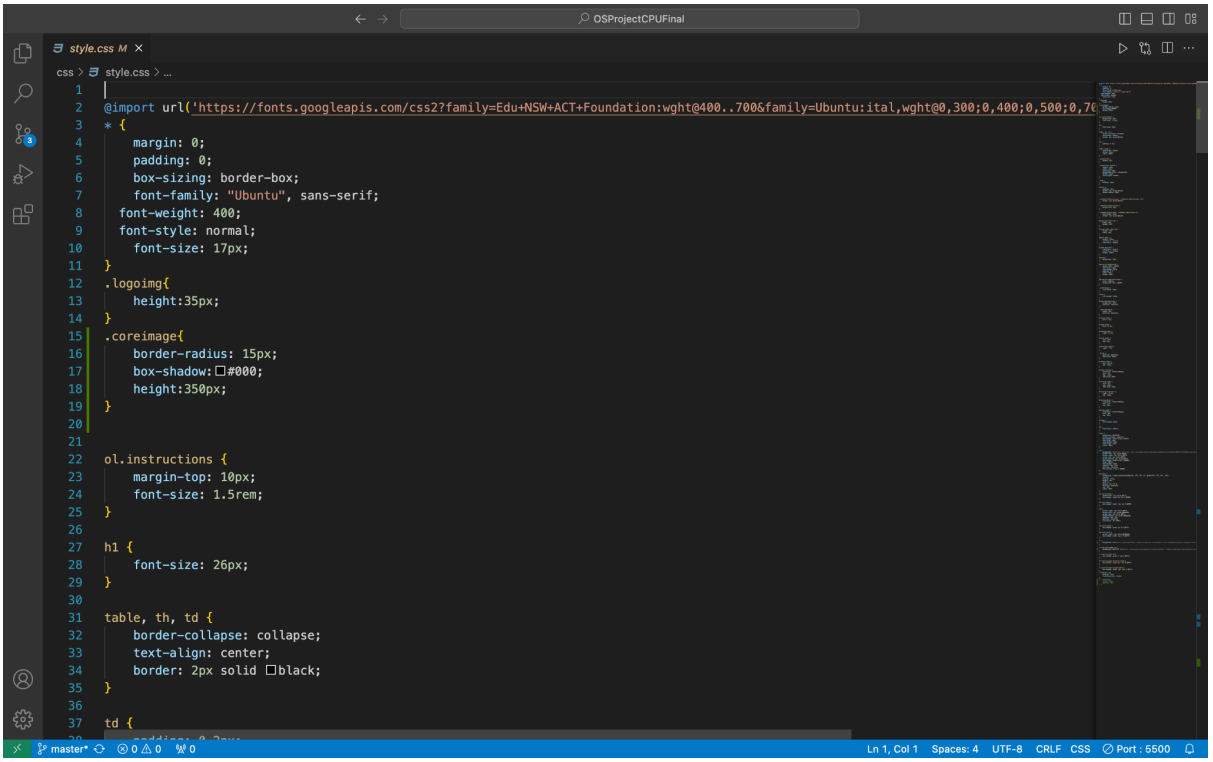

OSProjectCPUFinal
functions.js M
functions.js > ...
let priorityPreference = 1; //priority preferences change
document.getElementById("priority-toggle-btn").onclick = () => {
let currentPriorityPreference = document.getElementById("priority-preference").innerText;
if (currentPriorityPreference == "high") {
document.getElementById("priority-preference").innerText = "low";
} else {
document.getElementById("priority-preference").innerText = "high";
}
priorityPreference *= -1;
};
let selectedAlgorithm = document.getElementById('algo');
//time quantum is included only for round robin algorithm and hidden for rest of them, the time for which a process can be
// executed at a time when the cpu is given to it.
function checkTimeQuantumInput() {
let timequantum = document.querySelector("#time-quantum").classList;
if (selectedAlgorithm.value == 'rr') {
timequantum.remove("hide");
} else {
timequantum.add("hide");
}
}
// to check the priority, if the process is preemptive or non preemptive
function checkPriorityCell() {
let prioritycell = document.querySelectorAll(".priority");
if (selectedAlgorithm.value == "pnp" || selectedAlgorithm.value == "pp") {
prioritycell.forEach((element) => {
element.classList.remove("hide");
});
} else {
prioritycell.forEach((element) => {
element.classList.add("hide");
});
}
}
//based on the algorithm selected the functions are executed
selectedAlgorithm.onchange = () => {
master*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript
Port : 5500

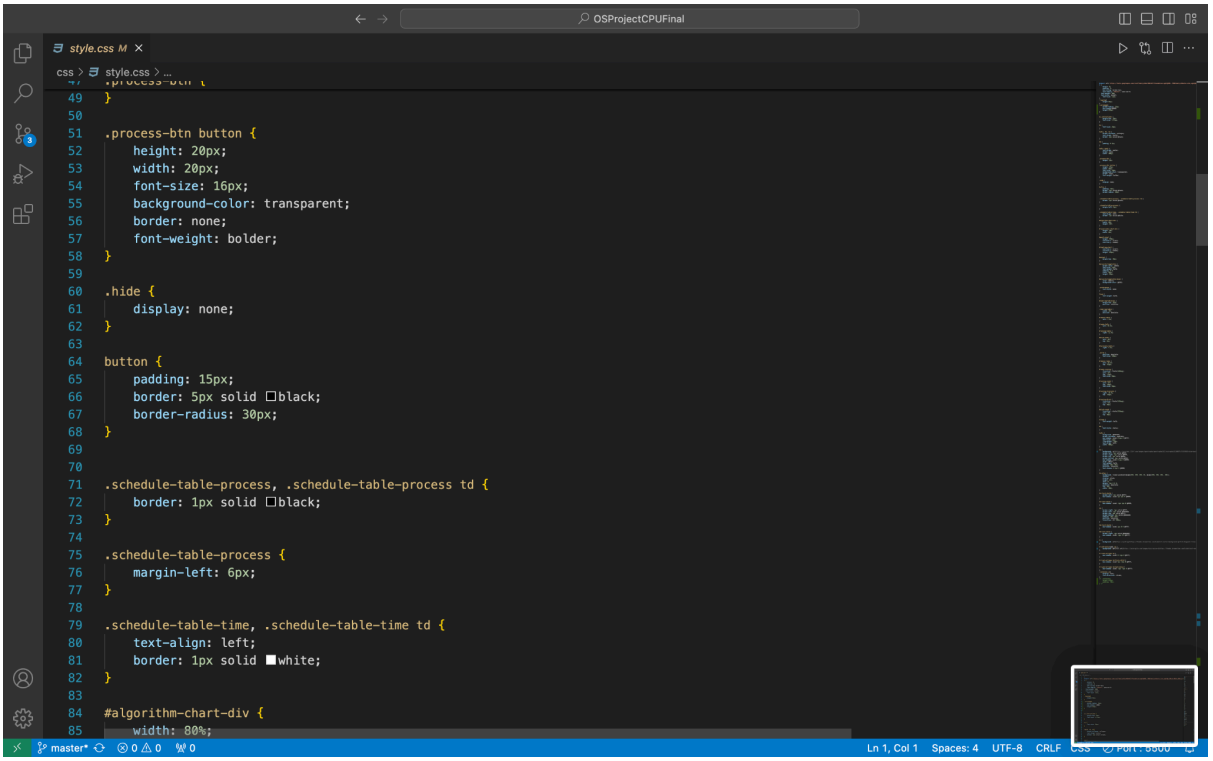

OSProjectCPUFinal
functions.js M
functions.js > ...
function inputOnChange() { //onchange EventListener for input
inputs.forEach((input) => {
input.onchange = () => {
input.value = inputVal;
}
} else //min 1 : time quantum, priority, process time
{
if (!isInt || (isInt && inputVal < 1)) {
input.value = 1;
} else {
input.value = inputVal;
}
}
}
}
});
}
inputOnChange();
let process = 1;
//resize burst time rows size on +/-
function gcd(x, y) {
while (y) {
let t = y;
y = x % y;
x = t;
}
return x;
}
function lcm(x, y) {
return (x * y) / gcd(x, y);
}
function lcmAll() {
let result = 1;
for (let i = 0; i < process; i++) {
master*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript

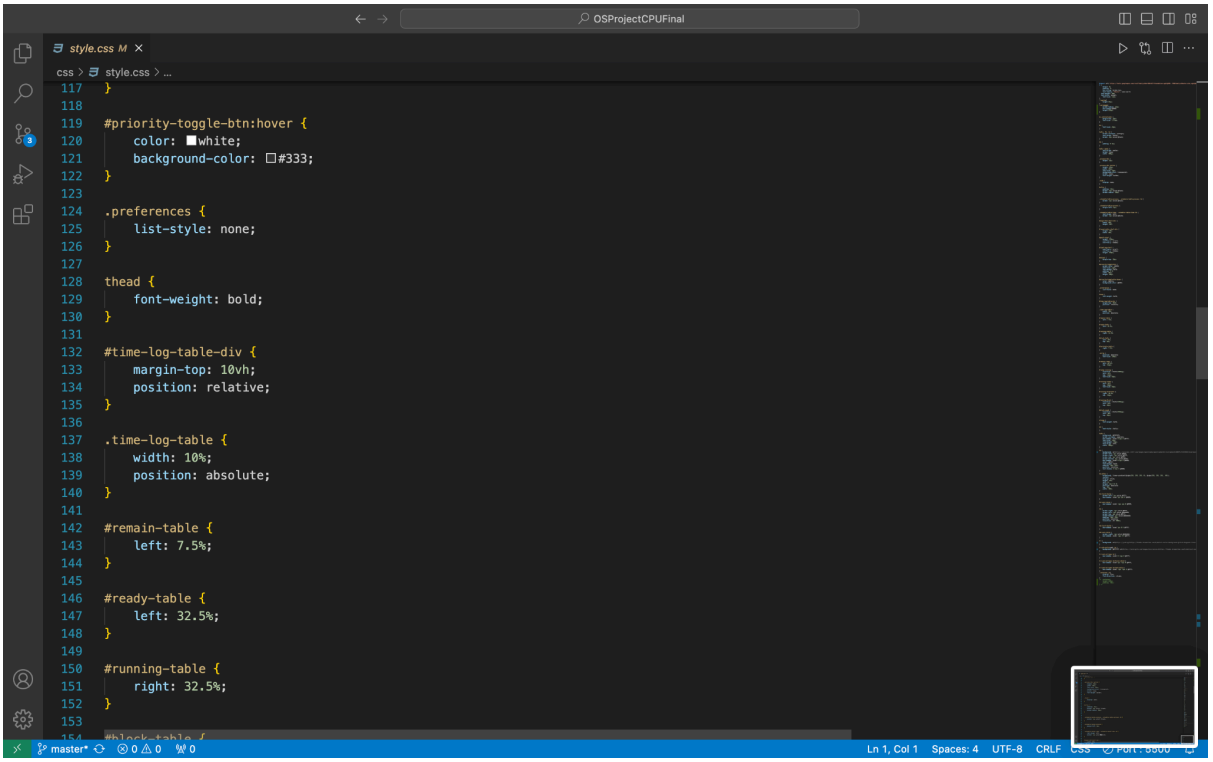

OSProjectCPUFinal
functions.js M
functions.js > ...
42 function inputOnChange() { //onchange EventListener for input
44 inputs.forEach((input) => {
46 input.onchange = () => {
54 input.value = inputVal;
55 }
56 } else //min 1 : time quantum, priority, process time
57 {
58 if (!isInt || (isInt && inputVal < 1)) {
59 input.value = 1;
60 } else {
61 input.value = inputVal;
62 }
63 }
64 }
65 }
66 });
67 }
68 inputOnChange();
69 let process = 1;
70 //resize burst time rows size on +/-
71
72 function gcd(x, y) {
73 while (y) {
74 let t = y;
75 y = x % y;
76 x = t;
77 }
78 return x;
79 }
80
81 function lcm(x, y) {
82 return (x * y) / gcd(x, y);
83 }
84
85 function lcmAll() {
86 let result = 1;
87 for (let i = 0; i < process; i++) {
master*
0 0
0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript

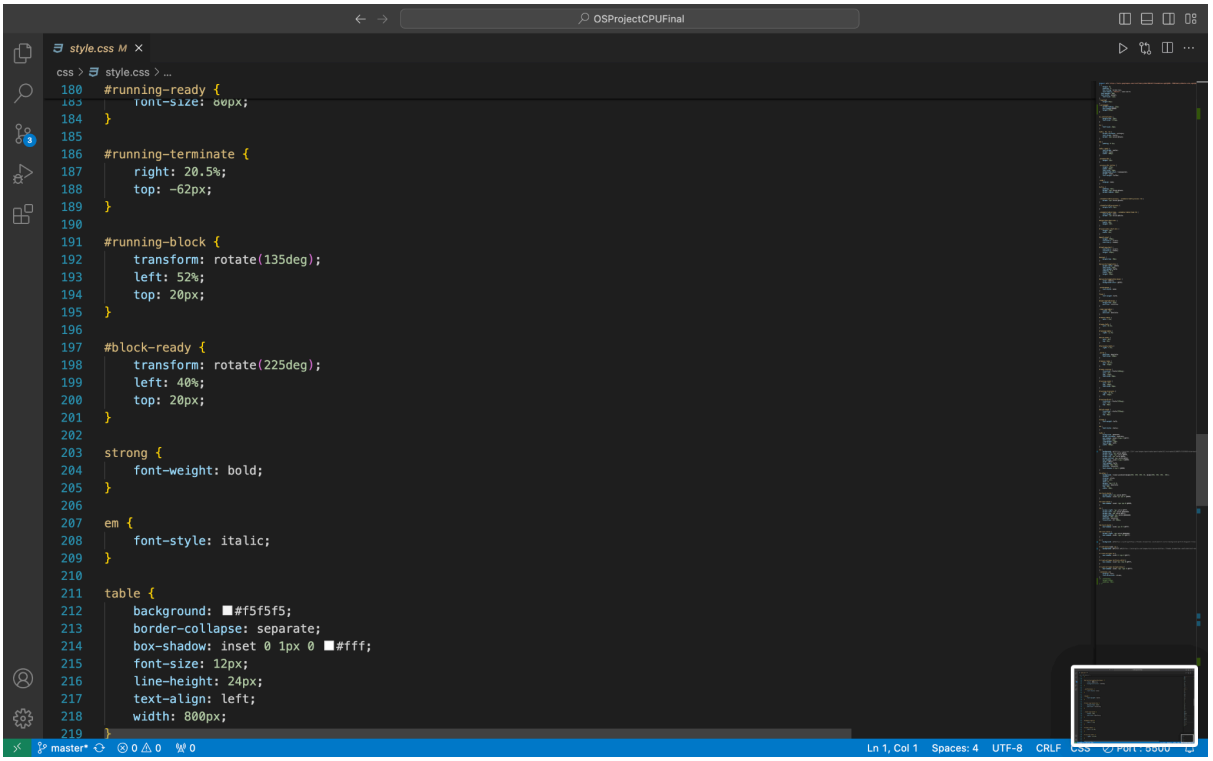

OSProjectCPUFinal
functions.js M
functions.js > ...
113 function addremove() { //add remove bt-io time pair add event listener
122 addbtns[i].onclick = () => {
139 newcell3.classList.add("cpu");
140 newcell3.classList.add("process-heading");
141 let newcell4 = row2.insertCell(processTimes[i] + 1);
142 newcell4.innerHTML = '<input type="number" min="1" step="1" value="1">';
143 newcell4.classList.add("process-time");
144 newcell4.classList.add("cpu");
145 newcell4.classList.add("process-input");
146 processTimes[i] += 2;
147 updateColspan();
148 inputOnChange();
149 };
150 }
151 let removebtns = document.querySelectorAll(".remove-process-btn");
152 for (let i = 0; i < process; i++) {
153 removebtns[i].onclick = () => {
154 if (processTimes[i] > 1) {
155 let table = document.querySelector(".main-table");
156 processTimes[i]--;
157 let row1 = table.rows[2 * i + 1];
158 row1.deleteCell(processTimes[i] + 3);
159 let row2 = table.rows[2 * i + 2];
160 row2.deleteCell(processTimes[i]);
161 processTimes[i]--;
162 table = document.querySelector(".main-table");
163 row1 = table.rows[2 * i + 1];
164 row1.deleteCell(processTimes[i] + 3);
165 row2 = table.rows[2 * i + 2];
166 row2.deleteCell(processTimes[i]);
167 updateColspan();
168 }
169 };
170 }
171 }
172 addremove();
173 //to add new process in the table
master*
0 0
0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript

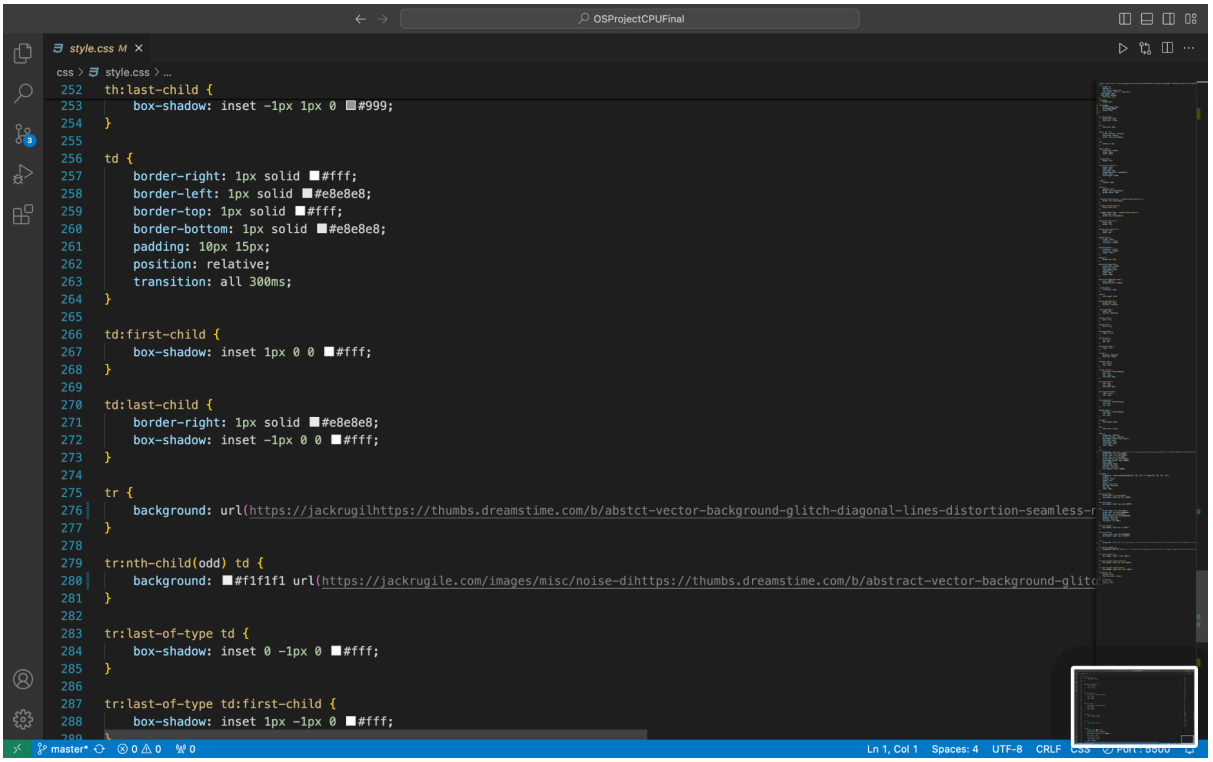

OSProjectCPUFinal
functions.js M
functions.js > ...
function CPUScheduler(input, utility, output) {
output.timeLog.push(JSON.parse(JSON.stringify(currentTimeLog)));
}
output.schedule.pop();
}
// function which calculates the output and displays it on screen
function calculateOutput() {
let outputDiv = document.getElementById("output");
outputDiv.innerHTML = "";
let mainInput = new Input();
let mainUtility = new Utility();
let mainOutput = new Output();
setInput(mainInput);
setUtility(mainInput, mainUtility);
CPUScheduler(mainInput, mainUtility, mainOutput);
setOutput(mainInput, mainOutput);
outputAverageTimes(mainOutput);
showOutput(mainInput, mainOutput, outputDiv);
}
//goes here first on clicking calculate
document.getElementById("calculate").onclick = () => {
calculateOutput();
};
master*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript


OSProjectCPUFinal
style.css M
css > style.css > ...
@import url('https://fonts.googleapis.com/css2?family=Edu+NSW+ACT+Foundation:wght@400..700&family=Ubuntu:ital,wght@0,300;0,400;0,500;0,7
* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: "Ubuntu", sans-serif;
font-weight: 400;
font-style: normal;
font-size: 17px;
}
.logoimg{
height:35px;
}
.coreimage{
border-radius: 15px;
box-shadow: #000;
height:350px;
}
ol.instructions {
margin-top: 10px;
font-size: 1.5rem;
}
h1 {
font-size: 26px;
}
table, th, td {
border-collapse: collapse;
text-align: center;
border: 2px solid black;
}
td {
master*
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
CSS
Port : 5500


OSProjectCPUFinal
style.css M
css > style.css > ...
}
.process-btn button {
height: 20px;
width: 20px;
font-size: 16px;
background-color: transparent;
border: none;
font-weight: bolder;
}
.hide {
display: none;
}
button {
padding: 15px;
border: 5px solid black;
border-radius: 30px;
}
.schedule-table-process, .schedule-table-process td {
border: 1px solid black;
}
.schedule-table-process {
margin-left: 6px;
}
.schedule-table-time, .schedule-table-time td {
text-align: left;
border: 1px solid white;
}
#algorithm-chart-div {
width: 80%;
master*
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF


OSProjectCPUFinal
style.css M
css > style.css > ...
}
#priority-toggle-btn:hover {
color: white;
background-color: #333;
}
.preferences {
list-style: none;
}
thead {
font-weight: bold;
}
#time-log-table-div {
margin-top: 10vh;
position: relative;
}
.time-log-table {
width: 10%;
position: absolute;
}
#remain-table {
left: 7.5%;
}
#ready-table {
left: 32.5%;
}
#running-table {
right: 32.5%;
}
master*
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF


OSProjectCPUFinal
style.css M
css > style.css > ...
#running-ready {
font-size: 80px;
}
#running-terminate {
right: 20.5%;
top: -62px;
}
#running-block {
transform: rotate(135deg);
left: 52%;
top: 20px;
}
#block-ready {
transform: rotate(225deg);
left: 40%;
top: 20px;
}
strong {
font-weight: bold;
}
em {
font-style: italic;
}
table {
background: #f5f5f5;
border-collapse: separate;
box-shadow: inset 0 1px 0 #fff;
font-size: 12px;
line-height: 24px;
text-align: left;
width: 800px;
}
master*
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF


OSProjectCPUFinal
style.css M
css > style.css > ...
th:last-child {
box-shadow: inset -1px 1px 0 #999;
}
td {
border-right: 1px solid #fff;
border-left: 1px solid #e8e8e8;
border-top: 1px solid #fff;
border-bottom: 1px solid #e8e8e8;
padding: 10px 15px;
position: relative;
transition: all 300ms;
}
td:first-child {
box-shadow: inset 1px 0 0 #fff;
}
td:last-child {
border-right: 1px solid #e8e8e8;
box-shadow: inset -1px 0 0 #fff;
}
tr {
background: url(https://jackrugilhttps://thumbs.dreamstime.com/b/abstct-vector-background-glitch-diagonal-lines-distortion-seamless-
}
tr:nth-child(odd) td {
background: #f1f1f1 url(https://jackrugile.com/images/misc/noise-dihttps://thumbs.dreamstime.com/b/abstract-vector-background-glitc
}
tr:last-of-type td {
box-shadow: inset 0 -1px 0 #fff;
}
tr:last-of-type td:first-child {
box-shadow: inset 1px -1px 0 #fff;
}
master*
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF