SIMULATION OF MAX-MIN SCHEDULING ALGORITHM

A project work submitted in partial fulfillment of the

requirements for the degree of

Bachelor of Science in Computer Science

to the

Periyar University, Salem– 11

Submitted By

S.S.SUBAHARINI

REG.NO:C23UG119CSC031

C.GOWTHAM

REG.NO: C23UG119CSC009

Under the guidance of

Dr. R.Barani, MCA., M.Phil.,Ph.D.,(UGC-NET)

Guest Lecturer, Department of Computer Science

DEPARTMENT OF COMPUTER SCIENCE

ARIGNAR ANNA GOVERNMENT ARTS COLLEGE,

(AFFILIATED TO PERIYAR UNIVERSITY,SALEM-11)

NAMAKKAL- 637002.

OCTOBER-2025

DECLARATION

I hereby declare that the project work entitled "**SIMULATION OF MAX-MIN SCHEDULING ALGORITHM"** Submitted to Department of Computer Science, Arignar Anna Government Arts College, Namakkal – 02 in partial fulfillment of the requirements for the award of degree of **BACHELOR OF COMPUTER SCIENCE** is a record of original work done by me under the direct supervision and guidance of **Dr. R.Barani, MCA., M.Phil.,Ph.D.,(UGC-NET).,** Guest Lecturer , Department of Computer Science, Arignar Anna Government Arts College, Namakkal -02.

Signature of the candidate

S.S.SUBAHARINI

REG.NO:C23UG119CSC031

C.GOWTHAM

REG.NO:C23UG119CSC009

DATE :

PLACE: Namakkal

Dr.R.Barani , MCA., M.Phil.,Ph.D.,(UGC-NET).,
Guest Lecturer,
Department of Computer Science,
Arignar Anna Government Arts College,
Namakkal -02.

CERTIFICATE

This is to certify that the project work entitled " **SIMULATION OF MAX-MIN SCHEDULING ALGORITHM"** submitted in partial fulfillment of the requirements of the degree of Bachelor of Science to the Periyar University, Salem is a record of Bonafide work carried out by **S.S.SUBAHARINI (**Reg.No:**C23UG119CSC031**) and **C.GOWTHAM** (RegNo: **C23UG119CSC009)** under my supervision and guidance.

Head of the Department Internal Guide

Date of Viva-voce :

Internal Examiner External Examiner

ACKNOWLEDGEMENT

At the outset, I would like to thank and honour god who gave me the wisdom and knowledge to complete this project.

I would like to profusely thank our college principal **Dr.M.Rajeshwari M.Sc., M.Phil.,Ph.D.,** Arignar Anna Government Arts College, Namakkal-02 for her encouragement and motivation to successfully carry out my project in house.

I express my gratitude to**Dr. P. KAMALAKKANNAN, MCA.,Ph.D.,** Associate professor and Head, Department of computer science, Arignar Anna Government Arts College, Namakkal-02 for being an unfailing source of inspiration and encouragement.

I would like to express my gratitude and sincere thanks to my guide **Dr.R.Barani, MCA., M.Phil., Ph.D.,(UGC-NET).,** Guest Lecturer, Department of computer science, Arignar Anna Government Arts College, Namakkal-02 for her source of investigation, guidance and suggestions throughout the project.

I express my sincere thanks to all other faculty members of computer science Department for thier support and help.

This acknowledgement will be incomplete without expressing my thanks to my parents and friends for their constant support and encouragement.

S.S.SUBAHARINI

C.GOWTHAM

TABLE OF CONTENTS

SYNOPSIS

The aim of this project is to implement and simulate the Max–Min Scheduling Algorithm to study how tasks can be scheduled efficiently. Task scheduling is important because it decides the order of execution, which directly affects waiting time, turnaround time, and overall system performance.

In this work, tasks are defined by their instruction volumes and resources are defined by their processing speeds. Execution times are calculated, and the Max–Min algorithm is applied. The algorithm selects the task with the maximum of the minimum execution times and schedules it first, ensuring better balance between small and large tasks.

The project is developed using HTML, CSS, and JavaScript. Users can input task and resource details, view execution-time calculations, and seethe scheduling results. The system also calculates and displays key metrics such as completion time, waiting time, turnaround time, average values, and throughput, providing a clear understanding of how the Max–Min algorithm works in practice.

CHAPTER–1

INTRODUCTION

The Max–Min Task Scheduling Algorithm is a scheduling approach designed to improve task allocation in computing systems. Task scheduling plays an important role in determining the order of execution of multiple tasks, directly influencing performance measures such as waiting time, turnaround time, and throughput. Without proper scheduling, tasks may be delayed, resources may remain underutilized, and overall efficiency may be reduced.

The Max–Min algorithm operates by first identifying the minimum execution time of each task, and then selecting the task with the maximum among these minimum times for scheduling. By prioritizing larger tasks in this manner, the algorithm ensures a balanced execution order, prevents starvation of long tasks, and achieves better load distribution across resources. This method is particularly useful in environments where both short and long tasks coexist.

This project simulates the working of the Max–Min scheduling algorithm using HTML, CSS, and JavaScript. The system accepts task details, calculates execution times, and applies the scheduling logic. It then displays important performance metrics such as completion time, waiting time, turnaround time, and throughput. This provides a practical and simplified platform to understand how scheduling algorithms can improve efficiency in task management.

1.1 SYSTEM SPECIFICATION

1.1.1 HARDWARE SPECIFICATION

Processor (CPU) : Intel corei5-7500 CPU@3.40GHz

Primary Memory (RAM) : 4GB

Storage (Hard Drive) : 500GB

1.1.2 SOFTWARE SPECIFICATION

Operating System : Windows 11

Front End : HTML, CSS, JavaScript

Browser : Google Chrome

Editor/IDE : VS Code

Platform : Runs locally in browser

HTML (Hypertext Markup Language):

HTML is the standard markup language used to create and structure web pages. It defines elements such as headings, paragraphs, images, and links, which are then displayed in the browser. HTML forms the foundation of every website and provides the basic framework for web content, allowing developers to structure data in readable format.

Syntax

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Example</title>
</head>
<body>
<h1>Welcome to HTML</h1>
<p>This is a paragraph of text.</p>
</body>
</html>
```

CSS(Cascading Style Sheets):

CSS is a style sheet language used to control the design and presentation of HTML elements. It controls the layout, design, and visual appearance of HTML elements, such as fonts, colors, spacing, positioning, and responsiveness for different screen sizes. By separating content (HTML) from style(CSS), web developers can create more attractive and maintainable web pages.

Syntax

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Example</title>
<style>
    h1 {
        color: blue;
```

```
        font-size:30px;
    }
</style>
</head>
</html>
```

JAVASCRIPT:

JavaScript is a scripting language used to add interactivity and dynamic behavior to web pages. It can perform calculations, validate forms, handle user actions, and update content in real-time without reloading the page.

Syntax

```
<!DOCTYPEhtml>
<html>
<head>
<title>JavaScriptExample</title>
</head>
<body>
<h1>JavaScript Demo</h1>
<p id="demo"></p>
<script>
    leta=5,b=10; let
    sum = a + b;
    document.getElementById("demo").innerHTML= "The sumis:"+sum;
</script>
</body>
</html>
```

CHAPTER–2

SYSTEM STUDY

2.1 Existing system

2.1.1 Description

In the existing scheduling approaches, tasks are generally assigned in a straightforward manner without considering proper optimization techniques. These methods follow simple rules for execution, which may be suitable for small-scale situations but fail when the number of tasks increases or when tasks vary in execution time. As a result, the overall efficiency of the system is reduced.

2.1.2 Drawbacks

1. **Inefficient Utilization of Resources** – Tasks are not scheduled intelligently, leading to idle time for some resources while others are overloaded.

2. **High Waiting Time** – Certain tasks may wait unnecessarily long before execution, which affects system responsiveness.

3. **Low Throughput** – Since tasks are not managed optimally, fewer tasks arecompleted in the same period, reducing efficiency.

4. **Unfairness in Scheduling** – Some tasks get priority while others are delayed, resulting in unequal treatment and poor fairness.

5. **Scalability Issues** – As the number of tasks increases, traditional approaches struggle to maintain performance and become less effective.

6. **Lack of Performance Optimization** – No proper calculation is carried out for minimizing turnaround time, waiting time, or maximizing throughput, which are essential measures of performance.

2.2 Proposed System

2.2.1 Description

The proposed system makes use of the Max–Min Scheduling Algorithm to improve task scheduling and resource utilization. This approach focuses on selecting the task with the maximum execution time and assigning it to the resource that can finish it the earliest.By doing so, larger tasks are completed without being delayed indefinitely, while smaller tasks can also be executed efficiently. The system also calculates key performance metrics such as Waiting Time (WT), Turnaround Time (TAT), Completion Time (CT), and Throughput, ensuring better evaluation of overall performance.

2.2.2 Features

- **Effective Task Allocation**–Tasks are distributed intelligently according to their execution time and available resources, ensuring balanced scheduling.
- **Reduced Waiting Time** – Both long and short tasks are managed fairly, preventing unnecessary delays.
- **Improved Throughput**–The number of tasks completed within a given time increases, making the system more productive.
- **Performance Analysis**–The system measures WT, TAT, and Throughput to evaluate efficiency and identify improvements.
- **Fair Scheduling** – All tasks are considered in a balanced manner, avoiding starvation or unfair delays.
- **User-Friendly Interface**–Implemented with web technologies like HTML, CSS, and JavaScript for simple input, clear execution, and easy result visualization.

CHAPTER–3

SYSTEM DESIGN AND DEVELOPMENT

3.1 Max–Min Scheduling Algorithm

The Max–Min Scheduling Algorithm is a heuristic approach widely applied in task scheduling to enhance efficiency, fairness, and throughput. The primary idea of the algorithm is to prioritize tasks with the maximum execution time and allocate them to the resource that can complete them in the minimum possible time. This ensures that longer tasks, which are more likely to delay overall completion if postponed, are handled early without causing starvation of smaller tasks.

3.2 Working of the Algorithm:

The Max-Min Scheduling Algorithm is implemented using a simulator which use HTML, CSS and Java script. It is divided into 5 modules.

1. **Input Module**
   This module collects the number of tasks to be executed with their tasks instruction volumes. It also inputs the number of resources on which tasks are executed with their processing capabilities.

2. **Computation Module**
   In this module the execution time for each task on resource is calculated and build allocation table.
   - For each task–resource pair, the expected completion time is calculated by considering the execution time of the task on the resource and the current availability of the resource.
   - A completion time matrix is prepared that lists how long each task would take on each resource.

3. **Scheduling Module**
   - For every task, the algorithm identifies the resource that results in the minimum completion time.
   - This ensures that, if scheduled, the task would finish as early as possible.
   - Among all tasks, the one with the maximum execution time (based on the minimum completion times) is chosen for scheduling.
   - This step ensures that larger tasks are prioritized, preventing them from being postponed indefinitely.
   - The selected task is assigned to the corresponding resource that yields its minimum completion time.
   - The resource's ready time is then updated to reflect the new load.

4. **Result Module**
   Displays process details including burst, completion, turnaround and waiting times.

(a) Burst Time (BT):

- The Burst Time (also called Execution Time) is the total amount of CPU time or resource time required by a task to complete execution.
- It is usually given as an input parameter for scheduling algorithms.

$$\mathbf{BT_i = ET_i - ST_i}$$

Where:

- BT_i =Burst time of task i
- ET_i = End(Completion) Time of task i
- ST_i = Start time of task i

(b) Completion Time(CT):

- The time at which a task finishes execution.

$$\mathbf{CT_i = ST_i + BT_i}$$

Where:

- CT_i =Completion Time of task
- ST_i= Start Time of task
- BT_i=Burst Time of task

(c) Turn around Time(TAT):

- The total time taken from the submission of a task to its completion.

$$\mathbf{TAT_i = CT_i - AT_i}$$

Where:

- TAT_i=Turnaround Time of task
- CT_i =Completion Time of task
- AT_i=Arrival Time of task

(d) Waiting Time(WT):

- The total time a task spends waiting in the ready queue before execution.

$$\mathbf{WT_i = TAT_i - BT_i}$$

Where:

- WT_i=Waiting Time of task
- TAT_i=Turn around Time of task
- BT_i=Burst Time of task

5. Analysis Module

In this module the throughput and makespan are calculated to evaluate the performance of the system.

(a) Throughput:

- The number of tasks completed per unit of time.

$$\textbf{Throughput} = \frac{\textbf{Number of Tasks Completed}}{\textbf{Total Time Taken}}$$

Where:

- Number of Tasks Completed =Total tasks scheduled and finished
- Total Time Taken= Sum of execution span (can be last completion time)

(b) Makespan:

- Makespan is the total length of the schedule that is the maximum completion time among all tasks. It measures how long the entire set of tasks takes to finish.

$$\textbf{Makespan} = \max(CT_i)\forall_i$$

Where:

- CT_i = Completion Time of task
- Makespan represents the finishing time of the last task in the schedule

3.3 Advantages of Max–Min Scheduling Algorithm

1. Fair Scheduling of Large Tasks

- Ensures that long tasks are executed without being postponed indefinitely.

2. Efficient Resource Usage

- Allocates tasks based on resource capability, reducing idle time.

3. Improved Throughput

- Allows more tasks to be completed within the given time frame

3.4 Features

The Max–Min Scheduling Simulator includes the following features:

1. **Input Management**
   a. Resource Definition:
      - Allows users to specify the number of resources and their processing speeds.
   b. Task Definition:
      - Enables users to enter the number of tasks along with their instruction volumes.

2. **Execution Time Calculation**
   a. Resource Allocation Table:
      - Automatically calculates execution times for each task on every resource based on task volumes and resource speeds.
   b. Dynamic Table Generation:
      - Displays results in a clear tabular form at for easy comparison.

3. **Scheduling Algorithm Simulation**
   a. Max–Min Scheduling Implementation:
      - Executes the Max–Min algorithm to allocate tasks to resources efficiently.
   b. Step-by-Step Processing:
      - Tracks the selection of tasks ,allocation to resources, and updates execution times dynamically.
4. **Result Visualization**
   a. Gantt-like Results Table:
      - Displays task IDs, burst times, completion times, waiting times, and turn around times.
   b. Performance Metrics:
      - Automatically computes average turnaround time, average waiting time, throughput, and makespan.

5. **User-Friendly Interface**
   a. Interactive Inputs:
      - Users can input data step by step without needing programming knowledge.
   b. Responsive Layout:
      - Clean HTML and CSS design for easy navigation and better readability.

CHAPTER–4

SYSTEM TESTING

4.1 Testing

Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies' or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

4.1.1 UNIT TESTING

Unit testing is the process of testing individual units or components of a software system in isolation to ensure that each performs as intended. It focuses on verifying the correctness of small, specific sections of code, such as functions or methods, before they are integrated with other parts of the system. This testing helps in identifying and fixing errors at an early stage, thereby improving code quality, maintainability, and reducing the overall cost of debugging in later phases of development.

4.1.2 INTEGRATION TESTING

Integration testing is performed after unit testing to check the interaction between integrated modules or components of a software system. Its purpose is to ensure that combined units work together correctly and that data flows accurately between them. This testing helps uncover issues related to interface mismatches, data communication errors, and module dependency conflicts that may not be visible during unit testing. Integration testing confirms that the integrated modules function as a single, unified system.

4.1.3 VALIDATION TESTING

Validation testing is carried out to determine whether the developed software meets the needs and expectations of the user or customer. It ensures that the final product fulfills its intended purpose and complies with all specified requirements. This testing verifies the overall functionality, performance, and usability of the software from the user's

perspective. It emphasizes building the right product that serves its real-world purpose effectively and satisfies business goals.

4.1.4 VERIFICATION TESTING

Verification testing is a systematic process used to evaluate whether the software product is being developed according to its specified requirements, design documents, and standards. It ensures that each development phase is correctly implemented and that the product is built in the right way. This testing focuses on reviewing documents, code, and design to confirm that all steps of the development process align with the planned objectives before proceeding to the next stage.

4.1.5 ACCEPTANCE TESTING

Acceptance testing is the final phase of software testing conducted to evaluate the system's readiness for deployment and delivery to the customer. It determines whether the software meets all acceptance criteria and business requirements agreed upon by the client. This testing is usually performed by the end user or client to verify that the system operates as expected in real-world conditions. Successful completion of acceptance testing signifies that the software is ready for release.

4.1.6 INPUT AND OUTPUT TESTING

Input and output testing focuses on verifying the correctness of data entering and leaving the system. It ensures that all input values are validated properly, processed accurately, and produce the expected output results. This testing helps confirm that the system handles valid and invalid data appropriately and that output information is displayed or stored in the correct format. It plays an important role in maintaining data accuracy and system reliability.

CHAPTER – 5

CONCLUSION

The Max–Min Scheduling System provides an efficient method for studying task scheduling in distributed environments. By applying the Max–Min algorithm, tasks are allocated to resources in a way that reduces makespan, balances workload, and improves overall performance. The system automates key calculations such as execution time, waiting time, turnaround time, throughput, and makespan, ensuring accuracy and eliminating manual effort.

With a simple and interactive interface, the project demonstrates the importance of scheduling algorithms in optimizing resource utilization and improving system efficiency. The results generated help students and researchers clearly understand the impact of scheduling strategies on performance.

Overall, this project emphasizes the role of effective task scheduling in achieving better throughput, reduced waiting times, and fair resource allocation, making it a valuable contribution to the study of distributed and cloud computing.

BIBLIOGRAPHY

The following sources were referenced during the development of the Max–Min Scheduling Simulator:

1. Books and Articles:

- Operating System Concepts (10^{th} Edition) by Abraham Silberschatz, PeterB. Galvin, and Greg Gagne.
- Cloud Computing: Principles and Paradigms by Rajkumar Buyya, James Broberg, and Andrzej M. Goscinski.
- Distributed and Cloud Computing: From Parallel Processing to the Internet of Things by Kai Hwang, Geoffrey C. Fox, and Jack Dongarra.

2. Web Resources:

- Geeks for Geeks – Scheduling Algorithms in Operating System: https://www.geeksforgeeks.org/scheduling-algorithms-in-operating-system/
- Tutorials Point – Cloud Computing Basics: https://www.tutorialspoint.com/cloud_computing/index.htm
- Research Gate–Max–Min Scheduling in Cloud Computing: https://www.researchgate.net

3. Development Tools and Platforms:

- Visual Studio Code(VS Code)– Microsoft Corporation.
- HTML, CSS, and JavaScript–Web Technologies.
- Git Hub– For version control and project management.

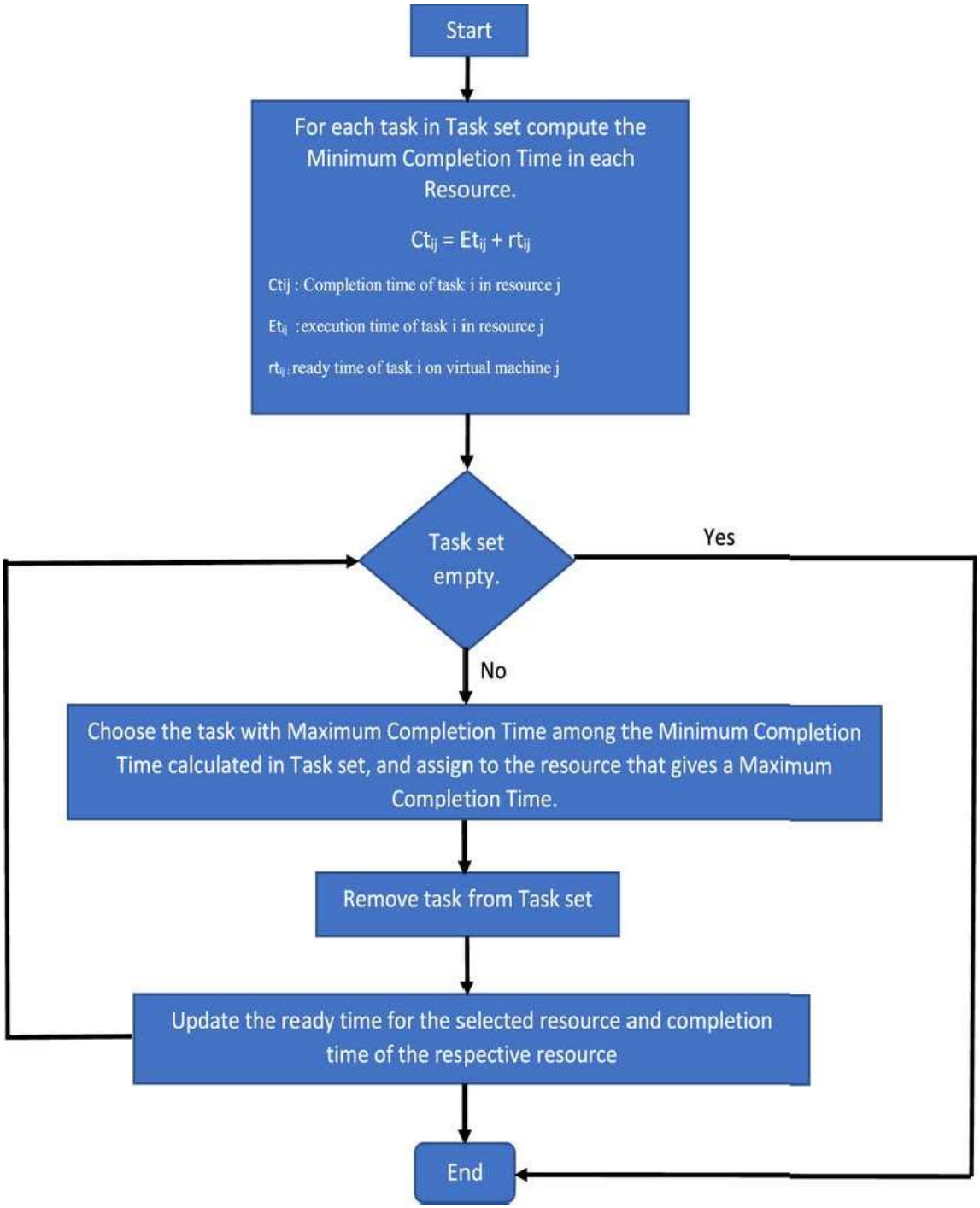
4. Additional References:

- Online forums and communities such as Stack Overflow and Git Hub Discussions for troubleshooting and practical code examples.
- YouTube tutorials for understanding scheduling algorithm implementation and simulation logic.

These resources were instrumental in ensuring the successful design, development ,and implementation of the Max–Min Scheduling Simulator.

APPENDICES

A. Flowchart Of Max-Min Algorithm



B. SAMPLECODING

```
<!DOCTYPEhtml>
<html lang="en">
<head>
  <metacharset="UTF-8">
  <metaname="viewport"content="width=device-width, initial-scale=1.0">
  <title>Max–MinSchedulingSimulator</title>
  <style>
    body{font-family:Arial,sans-serif;margin:20px;} h1 {
    text-align: center; }
    table{border-collapse:collapse;width:100%;margin-top:15px;}th,
    td { border: 1px solid #ccc; padding: 8px; text-align: center; }
    th{background:#f2f2f2;}
    input{ padding: 5px; width:120px; margin: 3px; }
    button{padding:8px15px;margin:10px0;background:#007bff;color:white;border: none;
border-radius: 5px; cursor: pointer; }
    button:hover{ background:#0056b3; }
    .result{ margin-top:20px;}
  </style>
</head>
<body>

<h1>Max–MinSchedulingSimulator</h1>

<label>NumberofResources:<inputtype="number" id="resources"></label>
<label>Numberof Tasks:<input type="number" id="tasks"></label>
<buttononclick="getSpeeds()">Next: EnterResourceSpeeds</button>

<div id="step2"></div>
<div id="step3"></div>
<div id="resourceTable"></div>

<div class="result">
  <h2>Results</h2>
  <table id="resultsTable">
    <thead>
      <tr>
        <th>ProcessID</th>
        <th>ArrivalTime</th>
        <th>BurstTime</th>
        <th>Completed Time</th>
        <th>WaitingTime</th>
        <th>Turnaround Time</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>

  <p>AverageTurnaroundTime:<inputid="avgTAT"readonly></p>
```

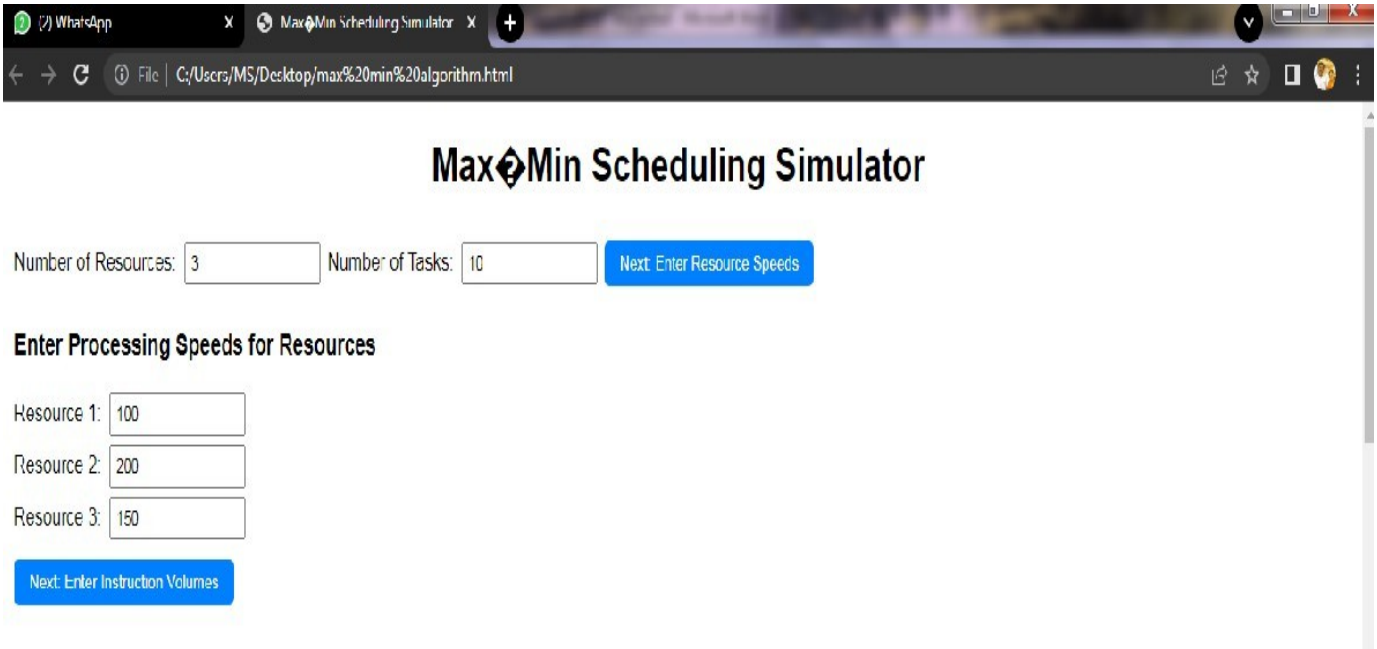
```
  <p>Average WaitingTime:<input id="avgWT"readonly></p>
  <p>Throughput:<input id="throughput"readonly></p>
  <p>Makespan:<inputid="makespan"readonly></p>
</div>

<script>
let nR, nT, speeds =[], volumes =[];

//Step2:Enterspeeds
functiongetSpeeds(){
  nR=parseInt(document.getElementById("resources").value); nT
  = parseInt(document.getElementById("tasks").value);

  lethtml="<h3>EnterProcessingSpeedsforResources</h3>"; for
  (let i = 0; i<nR; i++) {
    html+=`Resource${i+1}:<inputtype="number"id="speed-${i}" placeholder="Speed"><br>`;
  }
  html+=`<buttononclick="getVolumes()">Next:EnterInstructionVolumes</button>`;
  document.getElementById("step2").innerHTML = html;
}

//Step3:Entertaskvolumes
function getVolumes() {
speeds = [];
  for (let i = 0; i<nR; i++) {
    speeds.push(parseFloat(document.getElementById(`speed-${i}`).value));
  }

  lethtml="<h3>EnterInstructionVolumesforTasks</h3>"; for
  (let j = 0; j <nT; j++) {
    html+=`Task${j+1}:<inputtype="number"id="vol-${j}" placeholder="Volume"><br>`;
  }
  html+=`<buttononclick="buildResourceTable()">BuildResourceAllocation Table</button>`;
  document.getElementById("step3").innerHTML=html;
}

//Step4:Buildresourceallocationtable
function buildResourceTable() {
volumes = [];
  for (let j = 0; j <nT; j++) {
    volumes.push(parseFloat(document.getElementById(`vol-${j}`).value));
  }

  lettable="<h3>ResourceAllocationTable(Execution
Times)</h3><table><tr><th>Task</th>";
  for(leti=0;i<nR;i++)table+=`<th>R${i+1}</th>`; table
  += "</tr>";
```

```
  letexecTimes=Array.from({length:nR},()=>Array(nT).fill(0)); for
  (let j = 0; j <nT; j++) {
    table+=`<tr><td>T${j+1}</td>`;
    for (let i = 0; i<nR; i++) {
      execTimes[i][j]=volumes[j] /speeds[i];
      table +=`<td>${execTimes[i][j].toFixed(2)}</td>`;
    }
    table +="</tr>";
  }
  table +="</table>";
  table+=`<buttononclick='runMaxMin(${JSON.stringify(execTimes)})'>RunMax-Min
Scheduling</button>`;
  document.getElementById("resourceTable").innerHTML=table;
}

// Step 5: Run Max-Min Algorithm
functionrunMaxMin(execTimesJSON){
  //Convert backtonumbers
  letexecTimes=JSON.parse(JSON.stringify(execTimesJSON)).map(row=>row.map(Number));

  letdone=Array(nT).fill(false); let
  currentTime = 0;
  letresult =[];

  for(let k = 0;k <nT;k++){
    letmin=Array(nT).fill(Infinity); let
    ptr = Array(nT).fill(-1);

    for(letj=0;j<nT;j++){ if
      (done[j]) continue;
      for(let i= 0;i<nR;i++) {
        if(execTimes[i][j] <min[j]){
          min[j]=execTimes[i][j];
          ptr[j] = i;
        }
      }
    }

    let p1 = -1, maxMin = -1;
    for(letj=0;j<nT;j++){
      if(!done[j]&&min[j]>maxMin){
        maxMin = min[j];
        p1 = j;
      }
    }
    if (p1 ===-1) break;

    let burstTime = min[p1];
```

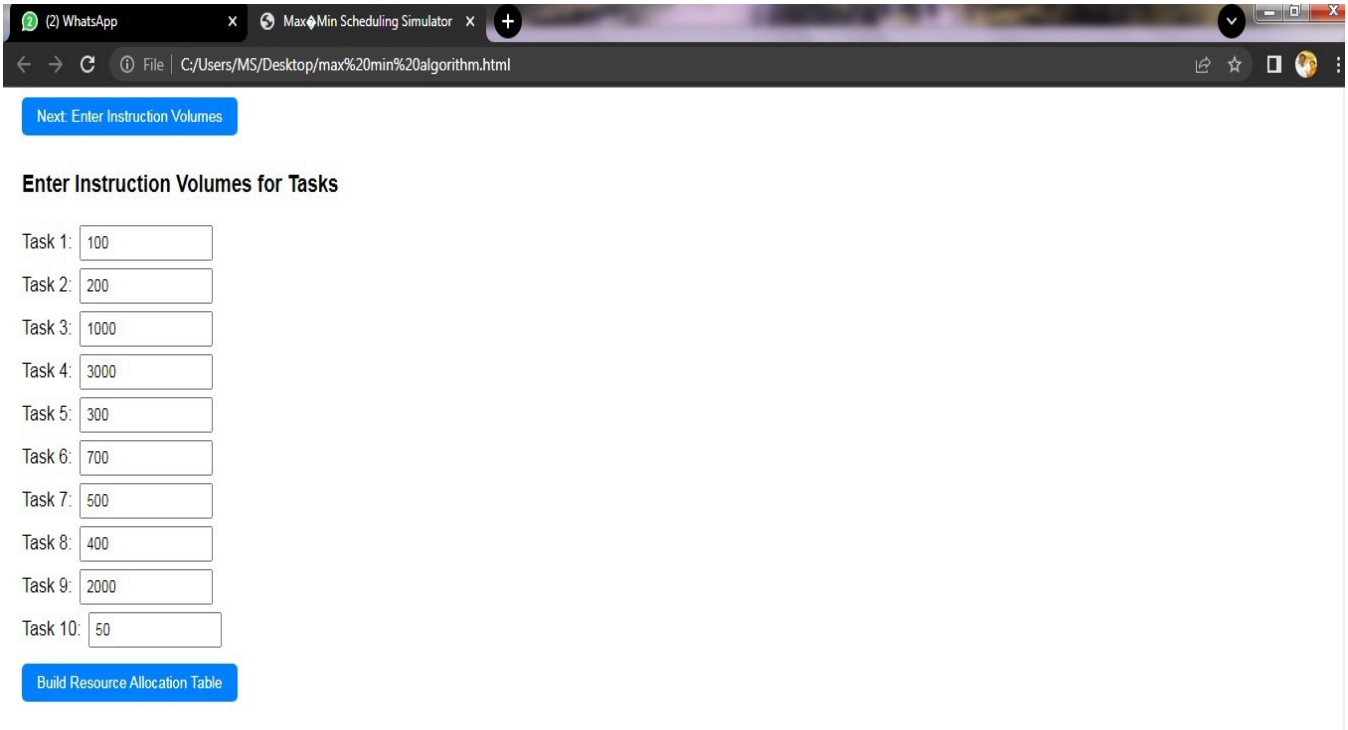
```
    currentTime+=burstTime;

    result.push({
      id: p1+1,
      arrival: 0,
      burst: burstTime.toFixed(2),
      completed:currentTime.toFixed(2),
      tat: currentTime.toFixed(2),
      wt:(currentTime- burstTime).toFixed(2)
    });
    done[p1]= true;

    for(let j = 0; j <nT; j++){
      if(!done[j])execTimes[ptr[p1]][j] +=burstTime;
    }
  }

  // Show final results
  lettbody=document.querySelector("#resultsTabletbody");
  tbody.innerHTML = "";
  lettotalTAT=0,totalWT=0,maxCT=0; for (let
  r of result) {
    tbody.innerHTML+=`<tr>
      <td>${r.id}</td><td>${r.arrival}</td><td>${r.burst}</td>
      <td>${r.completed}</td><td>${r.wt}</td><td>${r.tat}</td>
    </tr>`;
    totalTAT+=parseFloat(r.tat); totalWT
    += parseFloat(r.wt);
    if(parseFloat(r.completed)>maxCT) maxCT =parseFloat(r.completed);
  }
  document.getElementById("avgTAT").value = (totalTAT / nT).toFixed(2);
  document.getElementById("avgWT").value = (totalWT / nT).toFixed(2);
  document.getElementById("throughput").value = (nT / maxCT).toFixed(2);
  document.getElementById("makespan").value= maxCT.toFixed(2); // ✅Added
}
</script>

</body>
</html>
```

C. Sample input & Sample Output

1. Resource speed Allocation



2. Task volume Allocation



3. Resource Allocation table

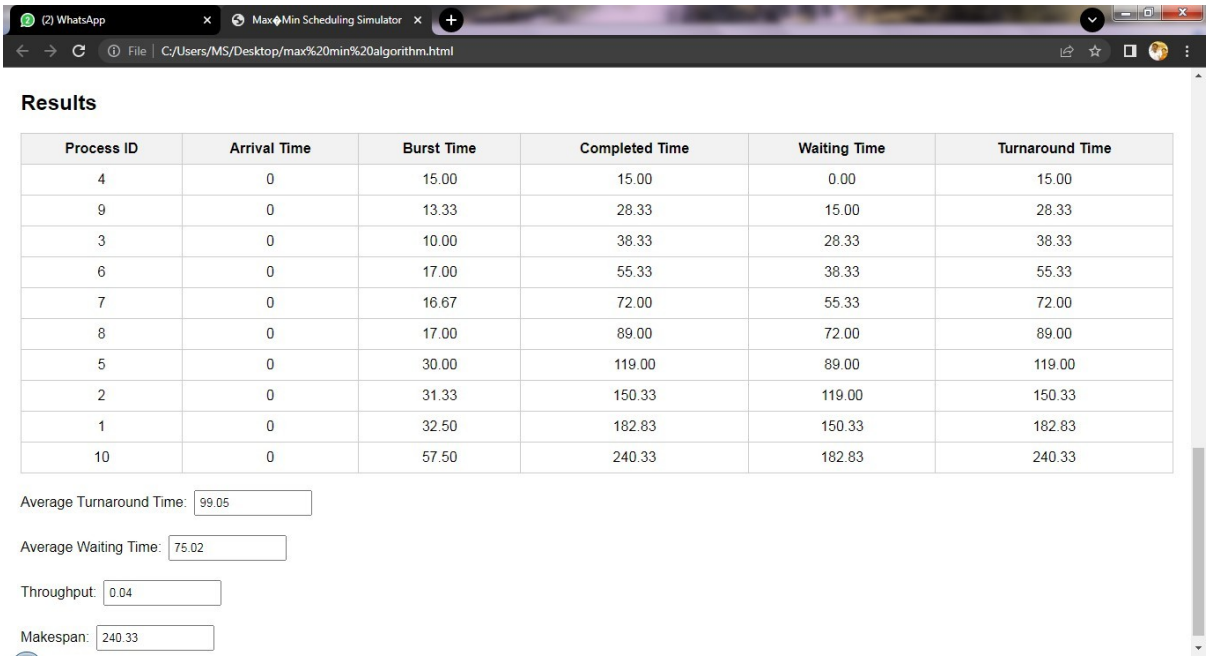
Build Resource Allocation Table

Resource Allocation Table (Execution Times)

Task	R1	R2	R3
T1	1.00	0.50	0.67
T2	2.00	1.00	1.33
T3	10.00	5.00	6.67
T4	30.00	15.00	20.00
T5	3.00	1.50	2.00
T6	7.00	3.50	4.67
T7	5.00	2.50	3.33
T8	4.00	2.00	2.67
T9	20.00	10.00	13.33
T10	0.50	0.25	0.33

Run Max-Min Scheduling

4. Max-Min Scheduling



Results

Process ID	Arrival Time	Burst Time	Completed Time	Waiting Time	Turnaround Time
4	0	15.00	15.00	0.00	15.00
9	0	13.33	28.33	15.00	28.33
3	0	10.00	38.33	28.33	38.33
6	0	17.00	55.33	38.33	55.33
7	0	16.67	72.00	55.33	72.00
8	0	17.00	89.00	72.00	89.00
5	0	30.00	119.00	89.00	119.00
2	0	31.33	150.33	119.00	150.33
1	0	32.50	182.83	150.33	182.83
10	0	57.50	240.33	182.83	240.33

Average Turnaround Time: 99.05

Average Waiting Time: 75.02

Throughput: 0.04

Makespan: 240.33