

1. INTRODUCTION

1.1 Introduction:

The word distributed in terms such as "distributed system", "distributed programming", and distributed algorithm originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other. Scheduling is the method by which work specified by some means is assigned to resources that complete the work. A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself. The concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU). The most common goal of scheduling is to minimize the expected runtime of a task set.

1.2 Project Background:

A distributed system consists of a set of heavy and light loaded systems (nodes). Scheduling variety of incoming tasks or jobs to the felicitous nodes, thus balancing load on these systems is an arduous issue. Load on a system or node can correspond to the queue length of tasks or processes that need to be processed.

“How can we execute a set of tasks T on a set of nodes N subject to some set of optimizing criteria C ?” is a problem often faced in scheduling. For the allocation of the resources efficient mechanisms are proposed [7]. These criteria can be minimizing the expected runtime of the tasks. Examples of such criteria can be minimizing the cost, minimizing the communication delay, giving priority to certain tasks which are considered important. Each

scheduling algorithm is better than others in its own way. So to schedule the tasks to the appropriate nodes we have to use the best one that furnishes the desirable results.

1.3 Existing Study:

The existing Novel Distributed scheduling model is an alternate to the centralized scheduling algorithms so far. In centralized scheduling the state of every computing resource are needed for routing. So it needs too much information which limits scalability. There are various centralized schemes proposed. Condor [4], a distributed job scheduler furnishes a queuing methodology, resource allocation modules for managing massive workloads. The distributed scheduling model [1], which depends on information available from CAN [6] (Content Addressable Network), offers a scalable routing. Kim et al [8] specifies a CAN overlay model used by Wave Grid. This approach is used to arrange the nodes systematically based on the resources which are currently unoccupied and piling up the length information of the corresponding queues.

The existing Idle Busy Policy [1] accepts at most one task in the queue. It has two states Idle and Busy. It also checks whether the execution nodes have enough memory to execute the assigned tasks.

The existing Make span Minimization Policy allocates tasks to the nodes which have shortest queues. These queues are sorted in First Come First Serve (FCFS) basis.

For both these policies task length, memory and disk space of the tasks to be submitted are needed. We are assuming the priority of the submitted tasks is not considered [2].

2. SOFTWARE PROJECT PLAN

2.1 Time Schedule For Various Phases:

Tasks Carried Out	Time Period
Initial Requirement Gathering	Dec 21-Jan 07
Gathering of complete requirement set	Jan 08-Jan 25
Learning and Literature Survey	Jan 26-Feb 05
Analysis and Design of methods	Feb 05-Feb 20
Implementation of Core Modules	Feb 21-Mar 20
Testing	Mar 20-Mar 30
Documentation	Mar 31-Apr 10

2.2 Team Member's Responsibility:

Member 1: Janani T.N

Enhancing the Idle Busy Policy by considering of the computational efficiency as a factor to improve the algorithm and implanting it effectively.

Member 2: Sushmita S

Enhancing the Make span minimization algorithm to further reduce the completion time of the tasks thus improving the algorithm.

2.3 Proposed System:

As we know, there are many algorithms which provide different ways of scheduling jobs. In our model there are 'n' fixed nodes. The values of these nodes are pre determined by variety of values which are stored and accessed by files. The capacity (C) for each of these nodes is calculated by the formula [3]

$$C_j = PEnum_j \times PEmips_j + VMbw_j$$

Where processing element, PEnum_j is the number processors in VM_j(node), PEmips_j is million instructions per second of all processors in VM_j and VMbw_j is the communication bandwidth ability of VM_j.

Node capacity is an important factor for allocating the tasks. It is the main factor which perceives whether the system node has the requirement and competence to accomplish the task requested by the user.

The number of tasks and the length of each task that are needed to be scheduled are taken in to the queue. These tasks are submitted by the user to the system. The tasks can be of varying length to understand the diversity of the algorithm. The length of each tasks are measured in FLOPS. Then the processing time of each task is calculated by the formula [3]

$$PT_i = TLi / Ci$$

Where PT_i is the processing time of each task, the TLi is the task length of the tasks. Based on the processing time values derived in the Enhanced Idle Busy, we allocate the tasks to the node which possess the least value. Thus the task is allocated to the node with the minimum processing time. Thus the first 'n' tasks are allocated to the 'n' nodes. The remaining tasks are allocated to the nodes with respect to memory and the number of instructions it can

execute per second (MIPS). The tasks are joined to the queue of the nodes which have the highest 'MIPS' and 'Memory' order. However one task is executed per node at a time, there by retaining the originality of the idle busy policy. The waiting times of the tasks executed by the nodes are calculated for each task and their average is found.

A comparison is done between the existing algorithm and our proposed one in which our algorithm produces better results. If two nodes have the same amount of computing efficiency as well as memory, we will allocate the task to the node on a FCFS (First Come First Serve) basis.

In the Enhanced Make span Minimization policy, the number of tasks and the length of each task are taken in the same manner as described above. These tasks are sorted in ascending order on the basis of their task length. The processing time of each task is calculated by the above formula. From the sorted tasks the first 'n' tasks are allocated to the first 'n' nodes and the remaining tasks are added to the nodes' queue in the reverse order. After the 'n' tasks are completed the next sets of tasks in the queue are executed.

Thus the queue will have one shortest job and one longest job. The completion times of the tasks executed by the nodes are calculated for each set and their average is found. A comparison is done between the existing algorithm and our proposed one in which our algorithm produces better results. This will reduce the make span of the tasks and thus strengthen the scheduling process.

2. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Functional Requirements:

Functional requirements include what the system should do i.e., the service provided for the user. The functional requirements of the system under study are as follows:

INPUT: The system or nodes gets its parameter by the use of files. A variety of files each with diverse values is given as input parameter to the nodes. The number of tasks and also the length of each task are submitted by the user. The user gets to decide which algorithm to choose IBP or MMP.

OUTPUT: Based on the chosen algorithm, the tasks are assigned to the most capable node that can reduce the completion time of the tasks.

3.2 Non-Functional Requirements:

3.2.1 Interface Requirements:

- ☐ Software Netbeans IDE, MS Access
- ☐ Version Netbeans:8.0

3.2.2 Operational Requirements:

The following are the minimum requirements that the target system is expected to possess for smooth running of this project.

- ☐ Operating Systems : Microsoft Windows (Version 8)
- ☐ Coding Language : Java

3.2.3 Resource Requirements:

The following are the minimum hardware requirements that the target system is expected to possess for running this program.

- ☐ Processor : Intel Core i3-3120M @ CPU 2.50 GHz
- ☐ RAM : 4.00 GB
- ☐ System Type : 32 bit OS

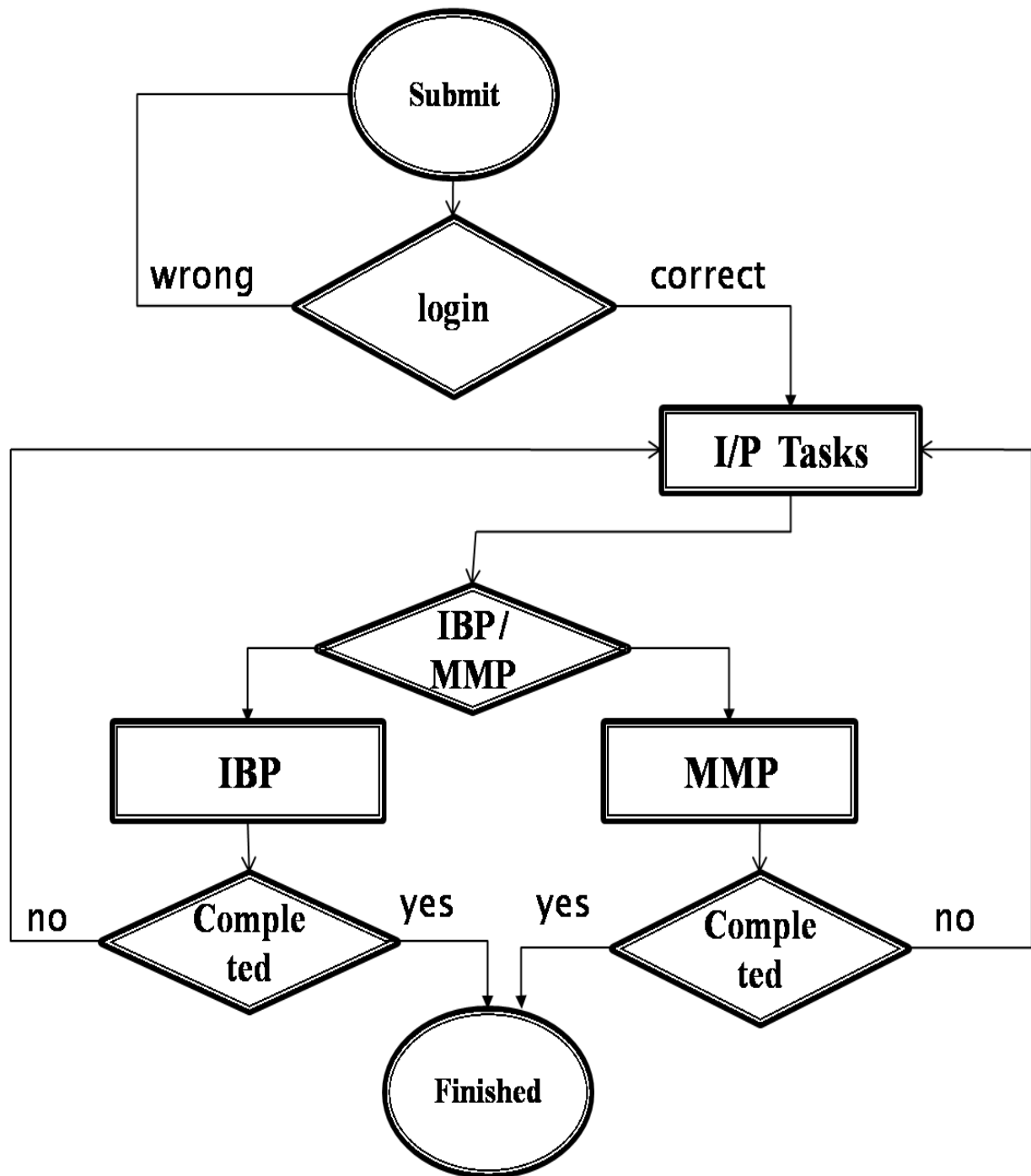
3.2.3. Quality and Reliability Requirements:

The project performs its required functions under stated conditions, which indicates its reliable nature.

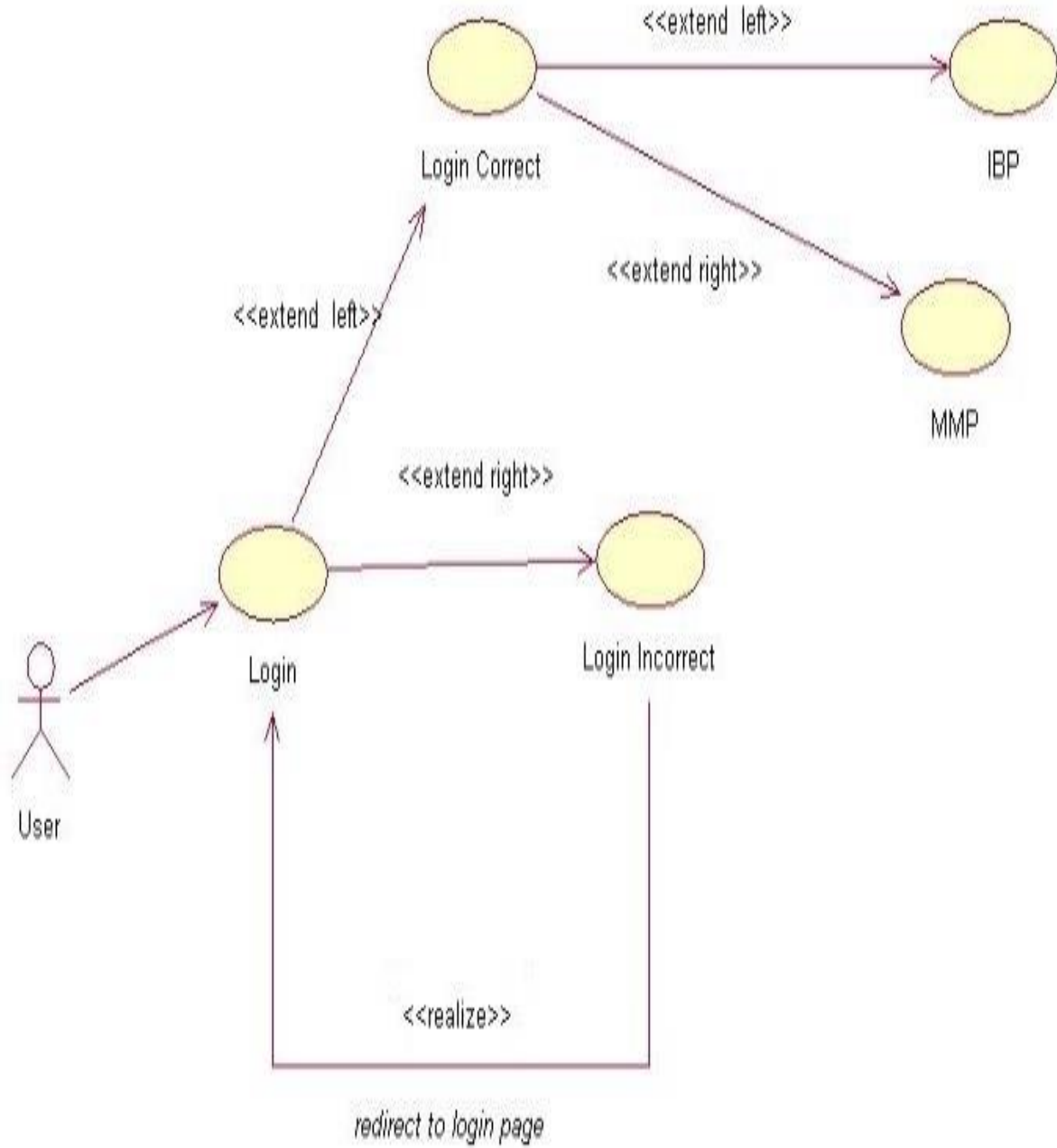
4. SYSTEM ANALYSIS

Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. System analysis is the process of examining the system's operation with a view of changing it to new requirements or cultivating its current working. System analysis involves collecting accurate data, understanding the process involved, recognizing problems and mentioning feasible suggestions for improving the system functioning. System should be analyzed with a diverse collection of data to check the efficiency of the algorithm. System analysis is an iterative process that continues until a preferred and suitable solution emerges. The result of this process is a logical system design. Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

4.1 Work Flow Diagram:



4.2 Use Case Diagram:



5. DESIGN

5.1 Front End Design:

We use Java as our programming language because Java has many boons over the other programming languages. Java is a set of several computer software products and specifications that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common, java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web on desktop computers.

One characteristic of Java is portability, which means that computer programs written in Java language must run similarly on any hardware/operating system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode. Java bytecode instructions are analogous to machine code. The heart of the Java platform is the concept of “virtual machine” that executes Java bytecode. This bytecode is the same no matter what hardware or operating system the program is running under. The use of bytecode as an immediate language permits java programs to run on any platform that has a virtual machine available. Java applets are programs that are embedded in other applications, typically in a web page displayed in a Web Browser.

Here we are using the Netbeans 8.0 as an execution environment. NetBeans is a software platform written in java. The NetBeans platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans

Platform, including the NetBeans Integrated Development Environment (IDE), can be extended by third party developers. The NetBeans Platform is a framework for simplifying the development of Java Swing desktop applications. NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types.

The front end consists of a user interaction “Login page” where the user submits his username and the password. We are validating whether the user submitted information is valid when user presses a ‘Submit’ button. If the information is correct the user gets to select by which routing algorithm his tasks needed to be routed (ie) by Enhanced Idle Busy Policy or by Enhanced Make span Minimization Policy. Then he can submit his tasks and their length so that they can be routed to the most suited execution node present in the environment.

5.2 Back End Design:

The back end is the MS Access database which has several users’ login identity. To connect with this database we use a driver called “ucanaccess” driver. We use queries to connect to the database through the driver. These identities comprises of each users’ username (the name by which they can login in to the system) and their password. The user enters their information on the Login page and clicks the ‘Submit’ button. The button has commands to verify whether the information given by the user matches that of the database.

6. CODING

6.1 Algorithm-IBP:

1. The text file which has parameter of a node is taken as an input.
2. The capacity is measured from the parameter's value.
3. The number of tasks and its length are accepted in a queue.
 - 3.1 For each task calculate the processing time
 - 3.2 If (number of nodes == number of tasks)
 - 3.2.1 Allocate each task to the node which has the minimum processing time.
 - 3.3 If (number of nodes < number of tasks)
 - 3.3.1 Allocate the first 'n' tasks to the 'n' nodes of least processing time.
 - 3.3.2 Allocate the remaining tasks based on the computational efficiency of the nodes.
4. Find the processing time of each task.
5. Repeat step 3 till all tasks are allocated.
6. Find the waiting time for each task and compute the average waiting time.

6.2 Algorithm-MMP:

- 1 The text file which has parameter of a node is taken as an input.
- 2 The CAPACITY is measured from the parameter's value.
- 3 The number of tasks and its length are accepted in a queue.
 - 3.1 Sort the queue in ascending order to arrange it in the order of its least task length.
 - 3.2 If (number of nodes == number of tasks)
 - 3.2.1 For each task calculate the processing time. Allocate each task to the node which has the minimum processing time.
 - 3.3 If (number of nodes < number of tasks)
 - 3.3.1 Allocate the first 'n' tasks to the 'n' nodes of least processing time.
 - 3.3.2 Allocate the remaining tasks by assigning the largest task length value to the shortest queue.
7. Find the processing time of each task.
8. Repeat step 3 till all tasks are allocated.
9. Find the completion time for each task and compute the average completion time.

6.3 Sample code:

```
package god;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class God {

    ResultSet rs;
    Statement st;
    Connection con;

    JFrame f = new JFrame("userlogin");
    JLabel label1 = new JLabel("username:");
    JLabel label2 = new JLabel("password:");

    JTextField text1 = new JTextField(15);
    JPasswordField text2 = new JPasswordField(15);
    JButton b1 = new JButton("SUBMIT");
    JPanel panel = new JPanel();
```

```

public God() {
    connect();
    frame();

}

public void connect() {
    try {

        String login =
"jdbc:ucanaccess://C:\\Users\\SYS\\Documents\\NetBeansProjects\\god\\j11.accdb";
        con = DriverManager.getConnection(login);

        st = con.createStatement();
    } catch (Exception ex) {
        System.out.println(ex);
    }

}

public void frame() {
    f.setSize(300, 300);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setVisible(true);

    panel.add(label1);
    panel.add(text1);
    panel.add(label2);
    panel.add(text2);
    panel.add(b1);
    f.add(panel);

    b1.addActionListener(new ActionListener() {

```



```

@Override
public void actionPerformed(ActionEvent e) {

    try {

        String u = text1.getText();
        String p = text2.getText();

        String sql = "select username,password from j11 where username='" + u + "'and
password='" + p + "'";
        rs = st.executeQuery(sql);
        int count = 0;
        while (rs.next()) {
            count = count + 1;
        }
        if (count == 1) {
            NextPage page = new NextPage();

            page.setVisible(true);

        } else if (count > 1) {
            JOptionPane.showMessageDialog(null, "Access Denied");
        } else {
            JOptionPane.showMessageDialog(null, "user not found");
        }

    } catch (Exception ex) {
    }

    });
}

public static void main(String[] args) {
    new God();
}

```

```

    }
}

package god;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileNotFoundException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

class NextPage extends JFrame {

    JFrame f1 = new JFrame("userlogin");

    JPanel panel1 = new JPanel();

    JButton b1 = new JButton("IBP");

    JButton b2 = new JButton("MMP");

    public NextPage() {

        frame();

    }
}

```

```

public void frame() {

    f1.setSize(300, 300);


    f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    f1.setVisible(true);


    panel1.add(b1);

    panel1.add(b2);

    f1.add(panel1);


    b1.addActionListener(new ActionListener() {

        @Override

        public void actionPerformed(ActionEvent e) {

            try {

                vm obj = new vm();

                obj.get();

                obj.met();

            } catch (FileNotFoundException ex) {

                Logger.getLogger(NextPage.class.getName()).log(Level.SEVERE, null, ex);

            }

        }

    });
}

```

```

b2.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        try {

            vm1 obj = new vm1();

            obj.get();

            obj.met();

        } catch (FileNotFoundException ex) {

            Logger.getLogger(NextPage.class.getName()).log(Level.SEVERE, null, ex);

        }

    }

});

}

}

```

```

package god;

```

```

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.util.Arrays;

```

```

import java.util.Scanner;

public class vm {

    float cpu[] = new float[5];

    float mips[] = new float[5];

    float bw[] = new float[5];

    float capacity[] = new float[5];

    float memory[] = new float[5];

    int k, l;

    Scanner sc = new Scanner(System.in);

    public void get() throws FileNotFoundException {

        int i, k;

        System.out.println("cpu is ");

        FileReader cpuu = new FileReader("cpu2.txt");

        Scanner sc0 = new Scanner(cpuu);

        while (sc0.hasNext()) {

            for (i = 0; i < 5; i++) {

```

```

        cpu[i] = sc0.nextFloat();

        System.out.println(cpu[i]);
    }

}

FileReader mipss = new FileReader("mips2.txt");

Scanner sc1 = new Scanner(mipss);

while (sc1.hasNext()) {

    for (i = 0; i < 5; i++) {

        mips[i] = sc1.nextFloat();

        System.out.println(mips[i]);

    }

}

System.out.println("BW IS ");

FileReader bww = new FileReader("bw2.txt");

Scanner sc2 = new Scanner(bww);

while (sc2.hasNext()) {

    for (i = 0; i < 5; i++) {

        bw[i] = sc2.nextFloat();

        System.out.println(bw[i]);

    }

}

System.out.println("BW IS ");

FileReader mem = new FileReader("memory2.txt");

Scanner sc3 = new Scanner(mem);

```

```

while (sc2.hasNext()) {

    for (i = 0; i < 5; i++) {

        memory[i] = sc3.nextFloat();

System.out.println(memory[i]);

    }

}

for (k = 0; k < 5; k++) {

    capacity[k] = cpu[k] * mips[k] + bw[k];

    System.out.println("Node Caapcity" + k + "::-" + capacity[k]);

}

}

public void met() {

    int index = 0;

    int zz;

    int i, j = 0;

    float wt[] = new float[10];

    System.out.println("enter the no of tasks :: ");

    int count = sc.nextInt();

    sc.nextLine();

    int tasklen[] = new int[count];

    float processtime[][] = new float[count][5];

```

```

System.out.println("enter length of each tasks ");

float[] a = new float[5];

float[] b = new float[5];

int[] allocated_order = new int[5];

for (i = 0; i < 5; i++) {

    tasklen[i] = sc.nextInt();


    System.out.println(" the task length " + i + ":: " + tasklen[i]);


    for (k = 0; k < 5; k++) {

        processtime[i][k] = (tasklen[i] / capacity[k]);

        a[k] = processtime[i][k];

        b[k] = processtime[i][k];

        System.out.println("process time " + k + ":: " + processtime[i][k]);

    }


    Arrays.sort(a);


    for (k = 0; k < 5; k++) {

        if (a[0] == b[k]) {

            j = k;

            break;

        }

    }

    allocated_order[i] = j;

```



```

        System.out.println("Task " + i + " is allocated to ::" + j);

        System.out.println("Node " + j + " is busy");

        capacity[j] = 0.00001f;

    }

    for (int aaa = 0; aaa < 5; aaa++) {

        System.out.println(allocated_order[aaa]);

    }

    int ab = 0;

    for (i = 5; i < count; i++) {

        System.out.println("enter length of each tasks ");

        tasklen[i] = sc.nextInt();

        float[] aa = new float[5];

        float[] bb = new float[5];

        float[] c = new float[5];

        System.out.println(" the task length " + i + " :: " + tasklen[i]);

        for (l = 0; l < 5; l++) {

            aa[l] = memory[l];

            bb[l] = memory[l];

            c[l] = mips[l];

        }

        Arrays.sort(aa);

```

```

for (l = 0; l < 5; l++) {

    if ((aa[l] == bb[l]) && c[l] == bb[l]) {

        j = l;

        break;

    }

    if ((aa[l] == bb[l]) || c[l] == bb[l]) {

        j = l;

        break;

    }

}

System.out.println("Task" + i + "is allocated to ::" + j);

System.out.println("Node " + j + " is busy");

memory[j] = memory[j] - ((225 * tasklen[i] * mips[j]) / 10);


for (zz = 0; zz < 5; zz++) {

    if (allocated_order[zz] == j) {

        index = zz;

        break;

    }

}

wt[ab] = processtime[index][j];

System.out.println("Waiting time :" + processtime[index][j]);


ab = ab + 1;

```

```

    }

    float sum = 0;

    for (int t = 0; t < 5; t++) {
sum = sum + wt[t];

    }

    System.out.println("sum is : " + sum);


    float avg = sum / 5;

    System.out.println("-----");

    System.out.println("Average wt is : " + avg);

    System.out.println("-----");

    System.out.println("-----END-----");

}

}

```

```
package god;
```

```
import java.io.FileNotFoundException;
```

```
import java.io.FileReader;
```

```
import java.util.Arrays;
```

```
import java.util.Scanner;
```

```
public class vm1 {
```

```

float cpu[] = new float[5];

float mips[] = new float[5];


float bw[] = new float[5];

float capacity[] = new float[5];

float capacity_new[] = new float[5];


float memory[] = new float[5];

int k, l;


Scanner sc = new Scanner(System.in);


public void get() throws FileNotFoundException {


    int i, k;

    System.out.println("cpu is ");

    FileReader cpuu = new FileReader("cpu2.txt");


    Scanner sc0 = new Scanner(cpuu);


    while (sc0.hasNext()) {


        for (i = 0; i < 5; i++) {


            cpu[i] = sc0.nextFloat();

```

```

        System.out.println(cpu[i]);

    }

}

FileReader mipss = new FileReader("mips2.txt");

Scanner sc1 = new Scanner(mipss);

while (sc1.hasNext()) {

    for (i = 0; i < 5; i++) {

        mips[i] = sc1.nextFloat();

        System.out.println(mips[i]);

    }

}

System.out.println("BW IS ");

FileReader bww = new FileReader("bw2.txt");

Scanner sc2 = new Scanner(bww);

while (sc2.hasNext()) {

    for (i = 0; i < 5; i++) {

        bw[i] = sc2.nextFloat();

        System.out.println(bw[i]);

    }

}

System.out.println("BW IS ");

FileReader mem = new FileReader("memory2.txt");

Scanner sc3 = new Scanner(mem);

while (sc2.hasNext()) {

```

```

    for (i = 0; i < 5; i++) {

        memory[i] = sc3.nextFloat();

        System.out.println(memory[i]);
    }

}

for (k = 0; k < 5; k++) {

    capacity[k] = cpu[k] * mips[k] + bw[k];

    System.out.println("Node Caapcity" + k + "::-" + capacity[k]);

}

for (int a = 0; a < 5; a++) {

    capacity_new[a] = capacity[a];

}

}

```

```

public void met() throws FileNotFoundException {

    int i, k, l;

    int ii = 0, ii1 = 0;

    int index;

    int j = 0, jj = 0;

    int taskno = 0;

    int i1 = 0;

    int x = 0, y = 0;

    int task_order1[] = new int[5];

    int task_order2[] = new int[5];

```

```

System.out.println("enter the no of tasks :: ");

int count = sc.nextInt();

sc.nextLine();


int tasklen[] = new int[count];

int tasklen1[] = new int[count];

int tasklen2[] = new int[count];

int tasklen_sort[] = new int[count];

float processtime[][] = new float[count][5];

float[] a = new float[5];

float[] b = new float[5];

int[] allocated_order = new int[5];

float completion_time[] = new float[5];


System.out.println("enter length of each tasks ");


FileReader tl = new FileReader("tasklength.txt");

Scanner s = new Scanner(tl);

while (s.hasNext()) {

    for (i = 0; i < count; i++) {

        tasklen[i] = s.nextInt();

        tasklen_sort[i] = tasklen[i];

        tasklen1[i] = tasklen[i];

        tasklen2[i] = tasklen[i];

        System.out.println(" the task length " + i + ":: " + tasklen[i]);
    }
}

```

```
}
```

```
Arrays.sort(tasklen_sort);
```

```
for (i = 0; i < count; i++) {
```

```
    System.out.println(tasklen_sort[i]);
```

```
}
```

```
for (i = 0; i < 5; i++) {
```

```
    for (l = 0; l < count; l++) {
```

```
        if (tasklen_sort[j] == tasklen1[l]) {
```

```
            tasklen1[l] = 0;
```

```
            taskno = l;
```

```
            task_order1[ii] = taskno;
```

```
            break;
```

```
        }
```

```
    }
```

```
    j++;
```

```
    ii++;
```

```
System.out.println("task no" + taskno);
```

```
for (k = 0; k < 5; k++) {
```

```
    processtime[taskno][k] = (tasklen[taskno] / capacity[k]);
```

```
    a[k] = processtime[taskno][k];
```

```
    b[k] = processtime[taskno][k];
```

```
    System.out.println("process time " + k + ":: " + processtime[taskno][k]);
```

```
}
```



```

Arrays.sort(a);

for (k = 0; k < 5; k++) {

    if (a[0] == b[k]) {

        jj = k;

        break;

    }

}

allocated_order[i] = jj;

System.out.println("Task " + taskno + " is allocated to ::" + jj);

System.out.println("Node " + jj + " is busy");

capacity[jj] = 0.00001f;

}

for (int aaa = 0; aaa < 5; aaa++) {

    System.out.println(allocated_order[aaa]);

}

int q;

int qu = 0;

float processtime1[][] = new float[count][5];

float aa[] = new float[5];

int zz = (count - 1);

```

```

for (i = 5; i < count; i++) {
    for (l = 0; l < count; l++) {
        if (tasklen_sort[zz] == tasklen2[l]) {
            tasklen2[l] = 0;
            taskno = l;
            task_order2[ii1] = taskno;
            break;
        }

    }

    zz--;
    ii1++;
    for (q = 0; q < 5; q++) {
        processtime1[taskno][q] = (tasklen[taskno] / capacity_new[q]);

        aa[q] = processtime1[taskno][q];

        System.out.println("process time " + q + ":: " + processtime1[taskno][q]);
    }

    jj = allocated_order[i1];
    i1++;

    System.out.println("Task " + taskno + " is allocated to ::" + jj);
    System.out.println("Node " + jj + " is busy");
    capacity_new[jj] = 0.00001f;

    System.out.println(processtime[task_order1[x]][jj]);

```

```

        System.out.println(processtime1[task_order2[x]][jj]);

        completion_time[y]          =          processtime[task_order1[x]][jj]          +
        processtime1[task_order2[x]][jj];

        System.out.println("-----");

        System.out.println("the completion time is" + completion_time[y]);

        System.out.println("-----");

        x++;

        y++;

    }

    float sum = 0;

    for (int t = 0; t < 5; t++) {

        sum = sum + completion_time[t];

    }

    System.out.println("sum is :" + sum);

    float avg = sum / 5;

    System.out.println("-----");

    System.out.println("Average wt is :" + avg);

    System.out.println("-----");

    System.out.println("-----END-----");

}

}

```

6.3 Experiments and Results:

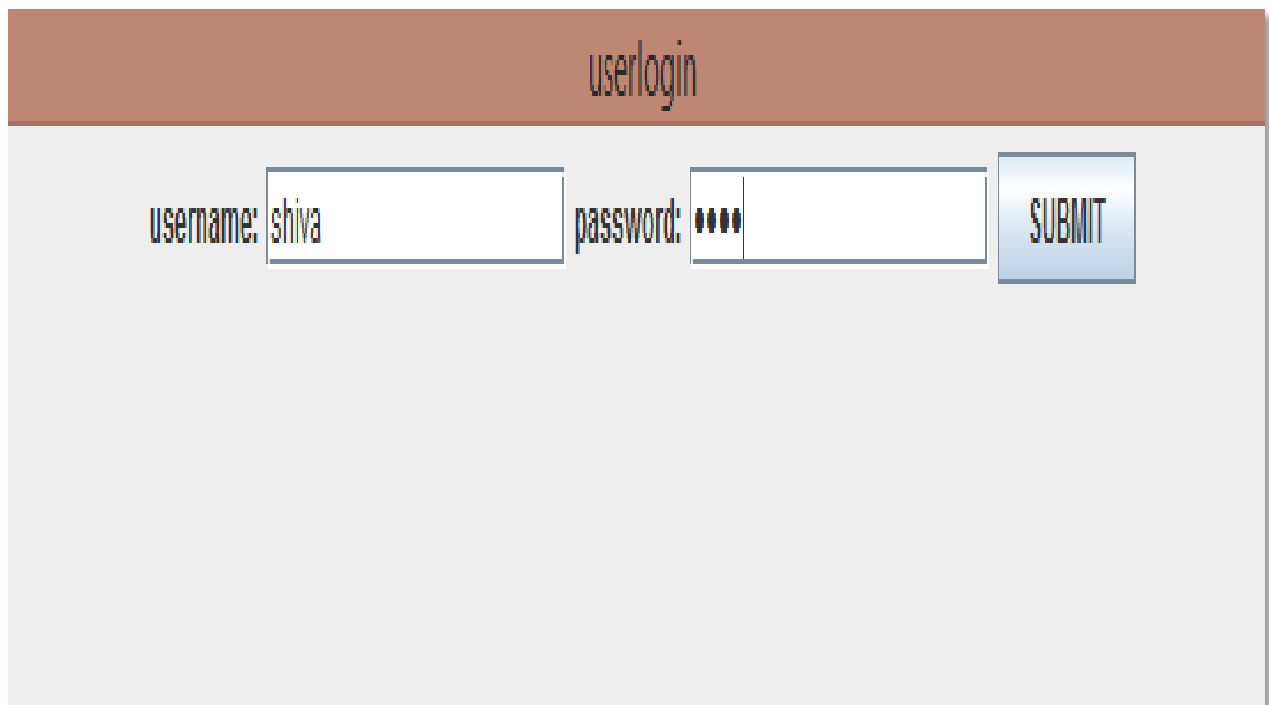
When different number of tasks and diverse task length values are given as input to a number of nodes both the algorithms appropriately chooses the node which reduces the waiting time of the tasks. The configuration of the nodes has also been changed and the algorithms are checked. For which both the algorithms produce efficient and desirable results which are shown in the testing. These results are compared with various existing models and it has been proved that our algorithm decreases the completion time of the tasks to a large extent than the other algorithms.

7. TESTING

7.1 Unit Testing:

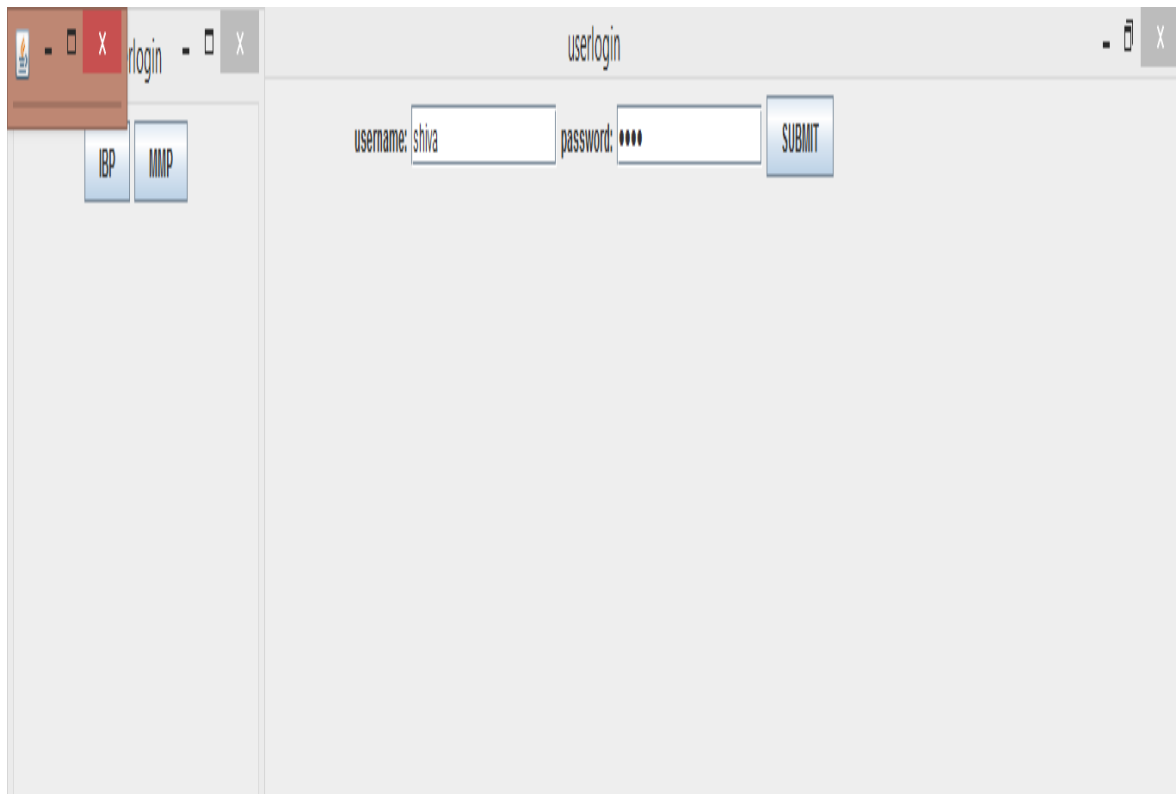
Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. A unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers.

7.1.1 Login with right username and password:

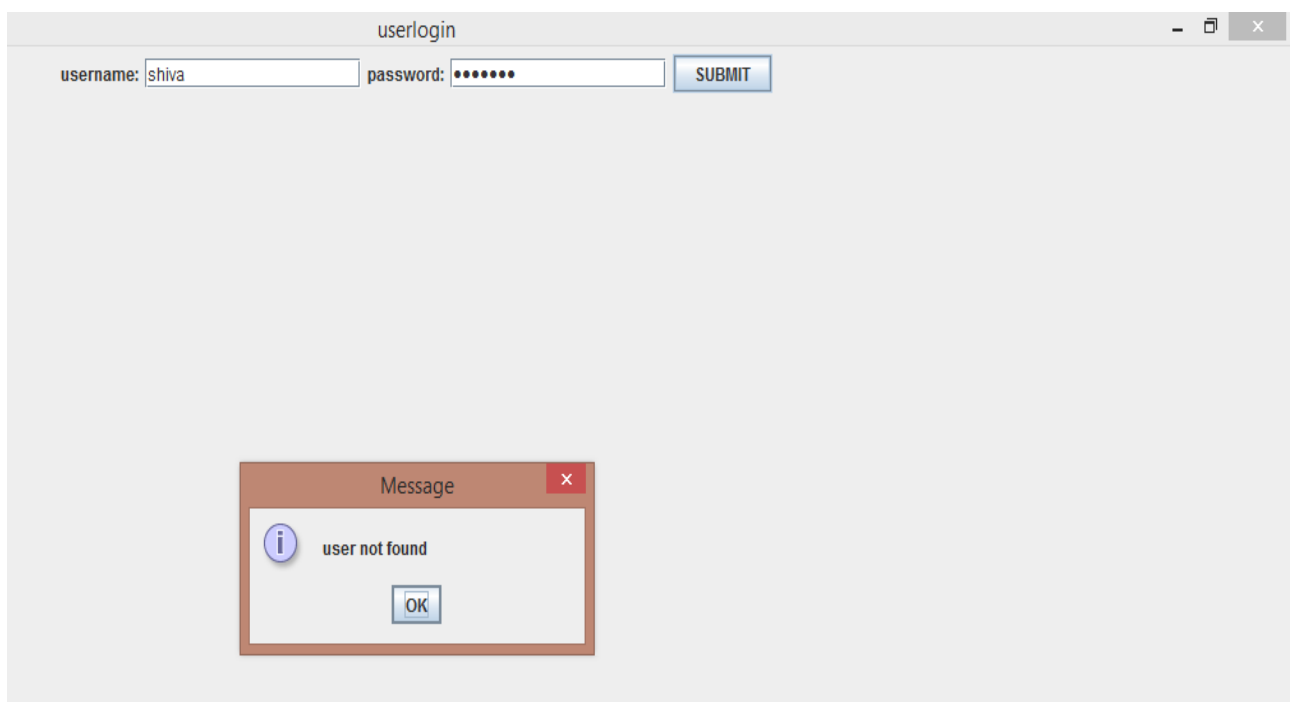


userlogin

username: shiva password: ●●●● SUBMIT



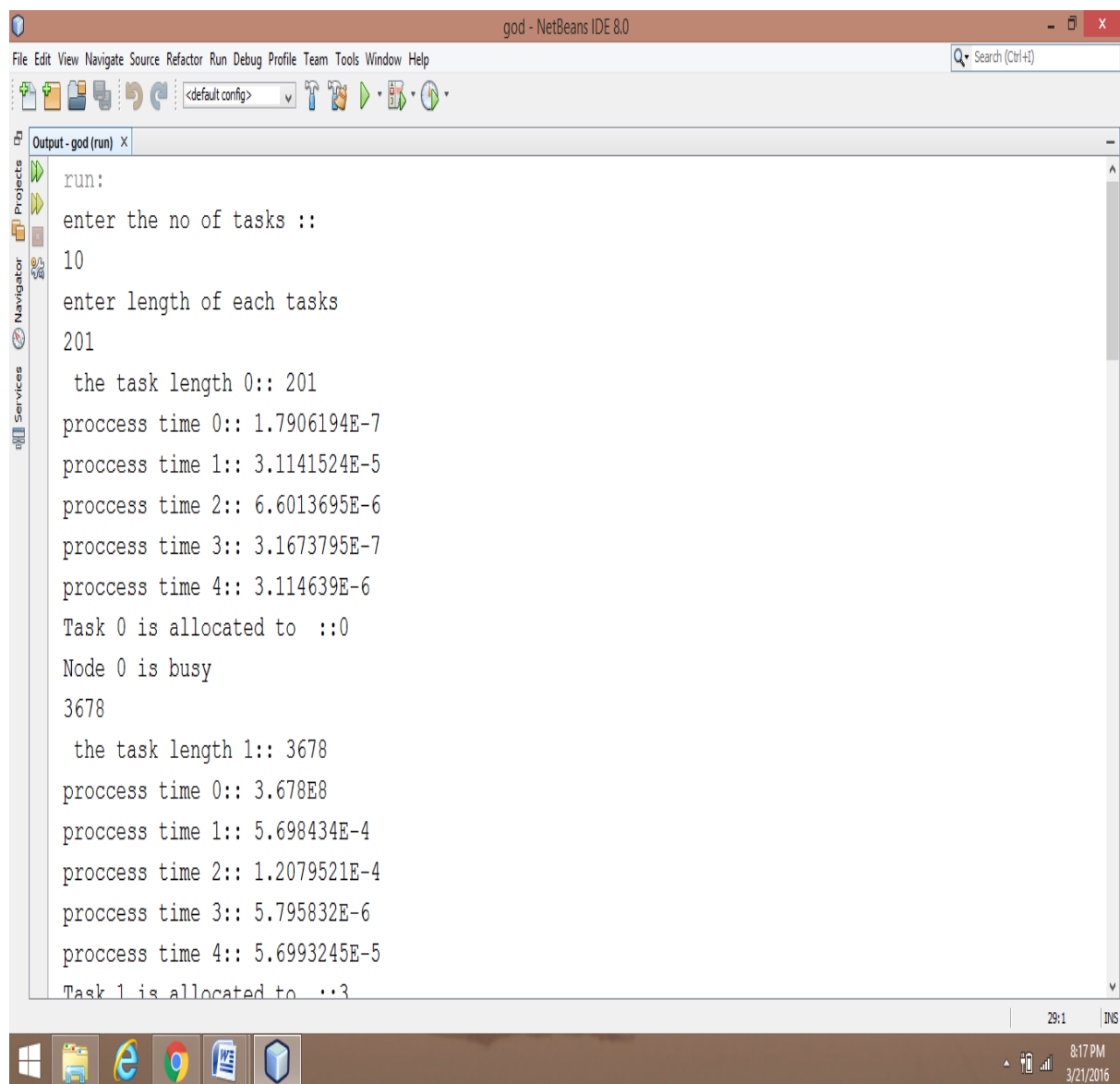
7.1.2 Login with incorrect username and password:



7.2 Validation Testing:

Validation testing is the evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process.

7.2.1 Testing of Enhanced Idle Busy Policy:



```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run) x
run:
enter the no of tasks ::
10
enter length of each tasks
201
    the task length 0:: 201
process time 0:: 1.7906194E-7
process time 1:: 3.1141524E-5
process time 2:: 6.6013695E-6
process time 3:: 3.1673795E-7
process time 4:: 3.114639E-6
Task 0 is allocated to ::0
Node 0 is busy
3678
    the task length 1:: 3678
process time 0:: 3.678E8
process time 1:: 5.698434E-4
process time 2:: 1.2079521E-4
process time 3:: 5.795832E-6
process time 4:: 5.6993245E-5
Task 1 is allocated to ::3
```

god - NetBeans IDE 8.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Output - god (run)

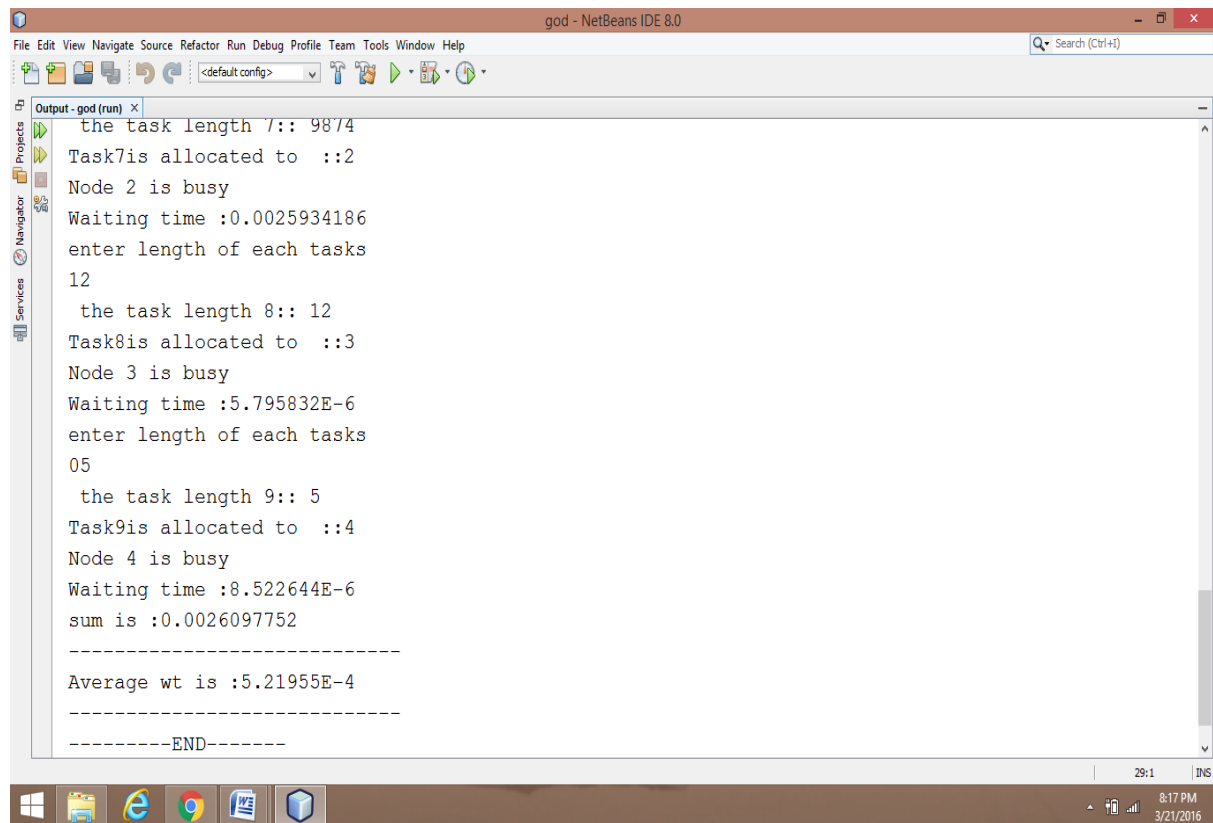
```
Task 1 is allocated to ::3
Node 3 is busy
550
    the task length 2:: 550
process time 0:: 5.5E7
process time 1:: 8.521312E-5
process time 2:: 1.8063449E-5
process time 3:: 5.5E7
process time 4:: 8.522644E-6
Task 2 is allocated to ::4
Node 4 is busy
78965
    the task length 3:: 78965
process time 0:: 7.8965002E9
process time 1:: 0.01223428
process time 2:: 0.0025934186
process time 3:: 7.8965002E9
process time 4:: 7.8965002E9
Task 3 is allocated to ::2
Node 2 is busy
12
```

29:1 | INS

8:17 PM
3/21/2016

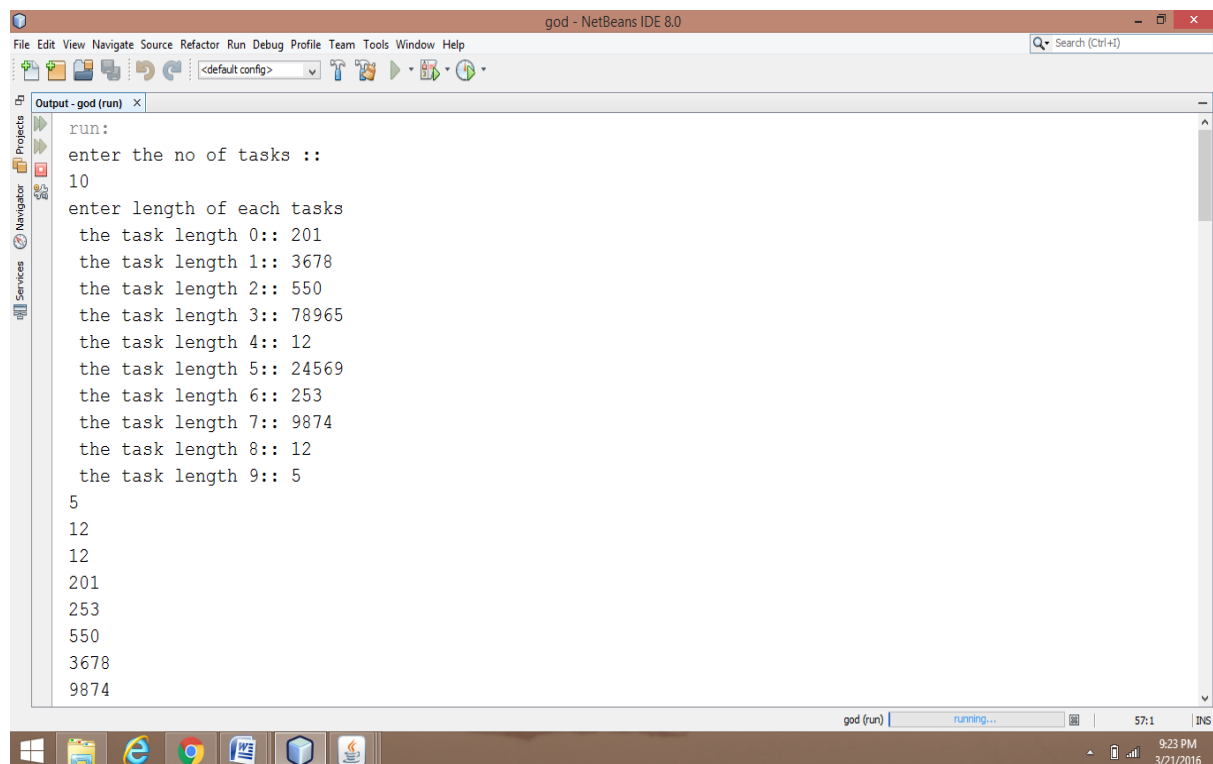

```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
the task length 4:: 12
proccess time 0:: 1200000.0
proccess time 1:: 1.8591954E-6
proccess time 2:: 1200000.0
proccess time 3:: 1200000.0
proccess time 4:: 1200000.0
Task 4 is allocated to ::1
Node 1 is busy
0
3
4
2
1
enter length of each tasks
24569
the task length 5:: 24569
Task5is allocated to ::0
Node 0 is busy
Waiting time :1.7906194E-7
enter length of each tasks
253
29:1 INS
8:17 PM 3/21/2016
```

```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
the task length 6:: 253
Task6is allocated to ::1
Node 1 is busy
Waiting time :1.8591954E-6
enter length of each tasks
9874
the task length 7:: 9874
Task7is allocated to ::2
Node 2 is busy
Waiting time :0.0025934186
enter length of each tasks
12
the task length 8:: 12
Task8is allocated to ::3
Node 3 is busy
Waiting time :5.795832E-6
enter length of each tasks
05
the task length 9:: 5
Task9is allocated to ::4
Node 4 is busy
29:1 INS
8:17 PM 3/21/2016
```



```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
the task length 7:: 9874
Task7is allocated to ::2
Node 2 is busy
Waiting time :0.0025934186
enter length of each tasks
12
the task length 8:: 12
Task8is allocated to ::3
Node 3 is busy
Waiting time :5.795832E-6
enter length of each tasks
05
the task length 9:: 5
Task9is allocated to ::4
Node 4 is busy
Waiting time :8.522644E-6
sum is :0.0026097752
-----
Average wt is :5.21955E-4
-----
-----END-----
29:1 INS
8:17 PM
3/21/2016
```

7.2.2 Testing of Enhanced Make span Minimization Policy:



```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
run:
enter the no of tasks ::
10
enter length of each tasks
the task length 0:: 201
the task length 1:: 3678
the task length 2:: 550
the task length 3:: 78965
the task length 4:: 12
the task length 5:: 24569
the task length 6:: 253
the task length 7:: 9874
the task length 8:: 12
the task length 9:: 5
5
12
12
201
253
550
3678
9874
god (run) running... 57:1 INS
9:23 PM
3/21/2016
```

```
24569
78965
task no9
process time 0:: 4.454277E-9
process time 1:: 7.746647E-7
process time 2:: 1.6421316E-7
process time 3:: 7.879054E-9
process time 4:: 7.747858E-8
Task 9 is allocated to ::0
Node 0 is busy
task no4
process time 0:: 1200000.0
process time 1:: 1.8591954E-6
process time 2:: 3.941116E-7
process time 3:: 1.8909729E-8
process time 4:: 1.859486E-7
Task 4 is allocated to ::3
Node 3 is busy
task no8
process time 0:: 1200000.0
process time 1:: 1.8591954E-6
process time 2:: 3.941116E-7
```

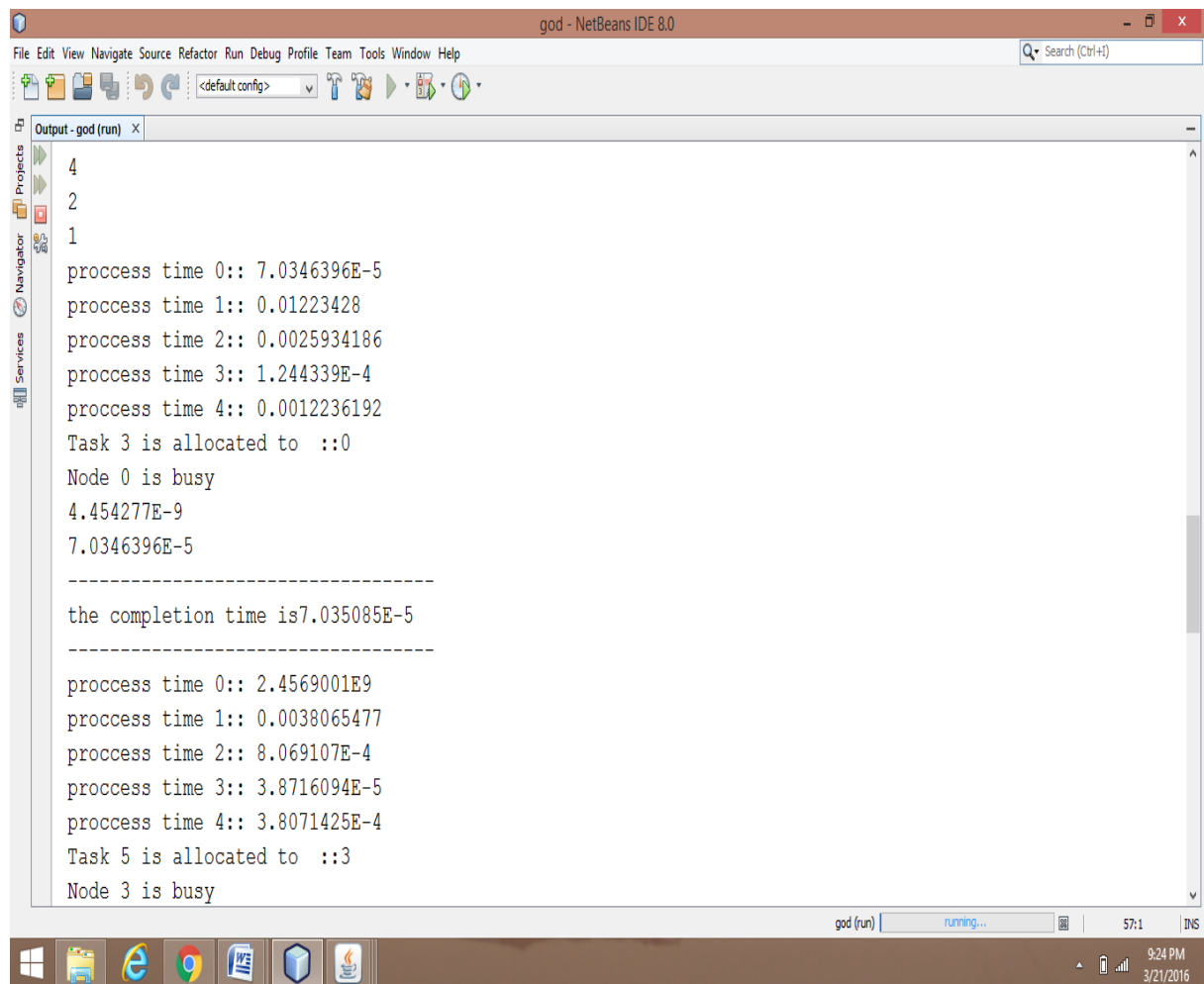
god (run) | running... | 57.1 | INS

9:23 PM
3/21/2016

```
process time 3:: 1200000.0
process time 4:: 1.859486E-7
Task 8 is allocated to ::4
Node 4 is busy
task no0
process time 0:: 2.01E7
process time 1:: 3.1141524E-5
process time 2:: 6.6013695E-6
process time 3:: 2.01E7
process time 4:: 2.01E7
Task 0 is allocated to ::2
Node 2 is busy
task no6
process time 0:: 2.53E7
process time 1:: 3.9198036E-5
process time 2:: 2.53E7
process time 3:: 2.53E7
process time 4:: 2.53E7
Task 6 is allocated to ::1
Node 1 is busy
0
3
```

god (run) | running... | 57.1 | INS

9:24 PM
3/21/2016



```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
1.8909729E-8
3.8716094E-5
-----
the completion time is3.8735005E-5
-----
proccess time 0:: 9.874E8
proccess time 1:: 0.0015298079
proccess time 2:: 3.2428818E-4
proccess time 3:: 9.874E8
proccess time 4:: 1.530047E-4
Task 7 is allocated to ::4
Node 4 is busy
1.859486E-7
1.530047E-4
-----
the completion time is1.5319065E-4
-----
proccess time 0:: 3.678E8
proccess time 1:: 5.698434E-4
proccess time 2:: 1.2079521E-4
proccess time 3:: 3.678E8
proccess time 4:: 3.678E8
god (run) running... 57:1 INS
9:24 PM 3/21/2016
```

```
god - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)
Output - god (run)
6.6013695E-6
1.2079521E-4
-----
the completion time is1.2739658E-4
-----
proccess time 0:: 5.5E7
proccess time 1:: 8.521312E-5
proccess time 2:: 5.5E7
proccess time 3:: 5.5E7
proccess time 4:: 5.5E7
Task 2 is allocated to ::1
Node 1 is busy
3.9198036E-5
8.521312E-5
-----
the completion time is1.2441116E-4
-----
sum is :5.1408424E-4
-----
Average wt is :1.0281685E-4
-----
-----END-----
god (run) running... 57:1 INS
9:24 PM 3/21/2016
```

8. IMPLEMENTATION

8.1 Problems Faced:

In the beginning, since we work with Netbeans 8.0 there is a slight problem of connecting to the MS Access database.

8.4 Lessons Learnt:

Various existing approaches for different scheduling algorithms were learnt and analyzed. A comparative study was carried out on the existing and new methods. We also came to know about the “ucanaccess driver” which is used instead of “sql driver” to connect to the MS Access database.

9. FUTURE PLANS

The proposed algorithms can be further updated by adding priority to the tasks that the user considers important. Also, we are assuming that the tasks do not have any specific deadlines. In the decentralized scheduling policies there is no fair sharing of platform. So we are planning to improvise our algorithm in these areas.

10. CONCLUSION

In the Enhanced Idle Busy Algorithm we provide an additional criterion by taking the computer efficiency into account. Thus we further improve the quality of the algorithm as well as reduce the waiting time of the tasks to a great extent.

In the Enhanced Make span Minimization Algorithm we decrease the make span of the tasks by allocating the largest tasks to the smallest queues. These algorithms can be used for scheduling tasks even in cloud environment.

11. REFERENCES

- [1] Javier Celaya , UnaiArronategui, "A task routing approach to large-scale scheduling", Future Generation Computer Systems (Sci-verse Science-direct), 2013.
- [2] Javier Celaya , Unai Arronategui " Fair scheduling of bag-of-tasks applications on large-scale platforms", Future Generation Computer Systems (Sci-verse Science-direct), 2015.
- [3] Dhinesh Babu L.D , P. Venkata Krishna " Honey bee behavior inspired load balancing of tasks in cloud computing environments", Applied Soft Computing (Sci-verse Science-direct), 2013.
- [4].T.Tannenbaum, D. Wright, K. Miller, M.Livny 2002, *Condor: a distributed job scheduler*, in: Beowulf Cluster Computing with Linux, pp. 307–350.
- [5].D.Anderson, BOINC (2004):*a system for public-resource computing and storage* ,in: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, pp. 4–10.
- [6].S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker (2006):*A scalable content addressable network*, in: *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM'01, ACM, pp. 161–172.
- [7]. K. Shyamala and T. Sunitha Rani(2015): An Analysis on Efficient Resource Allocation Mechanisms in Cloud Computing : Indian Journal of Science and Technology, Vol 8(9), 814–821.
- [8].J.-S. Kim, B. Nam, P. Keleher, M. Marsh, B. Bhattacharjee, A. Sussman (2008), *Tradeoffs in matching jobs and balancing load for distributed desktop grids*, Future Generation Computer Systems 24 415–424.

11.1. Web References

- <https://books.google.co.in/books>
- <http://stackoverflow.com/>
- <http://webdiis.unizar.es/~jcelaya/stars/>
- <https://forums.netbeans.org/ptopic28617.html>
- <http://www.coderanch.com/t/255551/Applets/java/Applet-Login-database>