Programme Name: **BCS HONS**

Course Code: **CSC 2624**

Course Name: **Distributed And Parallel Computing**

Assignment / Lab Sheet / **Project** / Case Study No. **1**

Date of Submission: **9/10/2021**

**Submitted By:**

Student Name: **Dipesh Tha Shrestha**

IUKL ID: **041902900028**

Semester: **Fourth Semester**

Intake: **September 2019**

**Submitted To:**

Faculty Name: **Manoj Gautam**

Department: **LMS**

1. **Develop a web based web server log analyzer tool that aggregate the information and display it in a visual dashboard**

**Answer:**
Before Developing a web based web server log analyzer tool that aggregate the information and display it in a visual dashboard, let us discuss about logs, log analysis, web log analyzer and web log analyzer tools.
Logs are one of the most valuable assets when it comes to IT system management and monitoring. As they record every action that took place on your network, logs provide the insight you need to spot issues that might impact performance, compliance, and security.

Log analysis is the process of making sense of computer-generated log messages, also known as log events, audit trail records, or simply logs. Log analysis provides useful metrics that paint a clear picture of what has happened across the infrastructure. You can use this data to improve or solve performance issues within an application or infrastructure. Looking at the bigger picture, companies analyze logs to proactively and reactively mitigate risks, comply with security policies, audits, and regulations, and understand online user behavior.

Log analysis tools are essential for effective monitoring, enabling you to extract meaningful data from logs and troubleshoot app- or system-level errors. They allow you to detect trends and patterns and use these insights to anticipate and mitigate risks and even guide your business decisions.

Web log analysis software (also known as a web log analyzer) is a type of web analytics software that parses a server log file from a web server and derives indicators about when, how, and by whom a web server is visited based on the values contained in the log file. Reports are typically generated immediately, but data extracted from log files can be stored in a database and used to generate various reports on demand.
Modern organizations track and log data for virtually all business processes, which is why web server log analysis tools are vital for effectively using this information to gain a clear picture of the state of your network. Event logs, security logs, transactions, web server uptime and CDN traffic are just a few of the types of logs you will collect.

So why do you need a web server log analyzer? All of this data can quickly become overwhelming. To maximize data utilization, you must be able to easily

find and analyze log files from a central location. Web log analysis tools allow you to filter logs, run live tail searches and query specific log data. These tools also present this data in a centralized view that can be accessed from anywhere.
In some cases, analysis tools will also provide log management capabilities. Instead of inspecting data after it's been recorded, log management deals with how you handle this data. This can mean creating policies for how you generate and store logs or automating when logs are archived after a certain amount of time.

## The Benefits of Log Analysis Tools

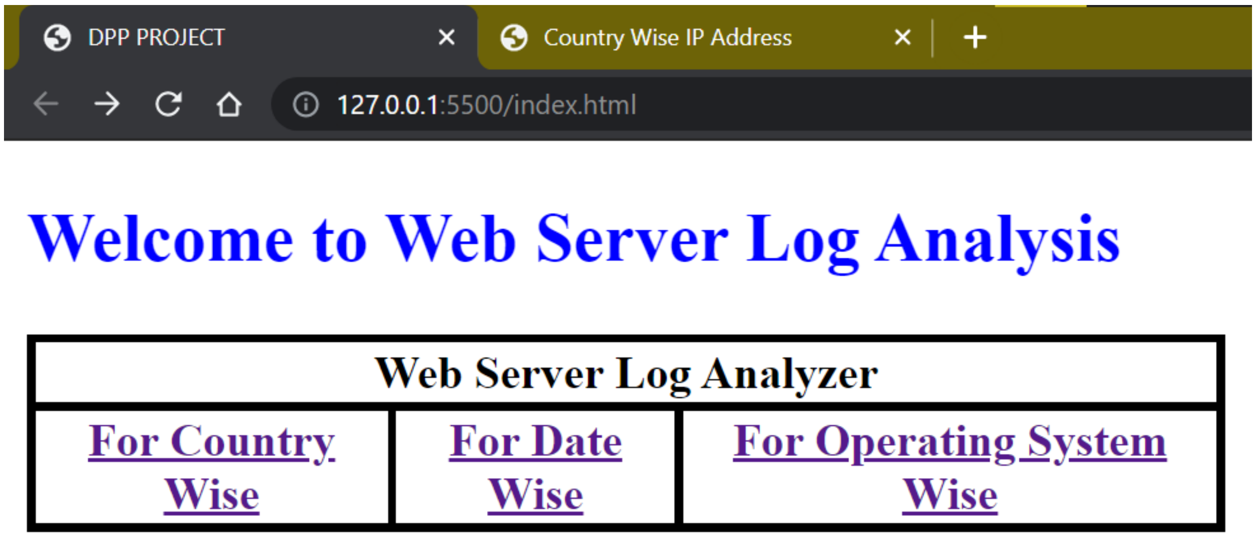
Effectively using the web log analysis tools can offer multiple benefits to make your job easy and ensure the efficiency of your network. Some of the benefits of log analysis tools include:

- You can identify key trends and anomalies to get a clear understanding of the health of your network and where there's room for improvement.
- You can correlate event data with downtime across your network to identify the root cause of issues and troubleshoot them quickly.
- Use these logs to identify unauthorized user access or suspicious activity to prevent data breaches and locate points of entry.
- With all of your log files in a single location, you will reduce the time you spend alternating between different log locations to find the information most relevant to the task at hand.

Here is the code **web server log analyzer tool homepage.**

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

<head>
   <title>DPP PROJECT</title>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link href="css/style.css" rel="stylesheet">
   <style>
table, td, th {
  border: 3px solid black;
}

table {
  width: 50%;
  border-collapse: collapse;
}
</style>
</head>

<body>
<h2 style="color:blue">Welcome to Web Server Log Analysis</h2>
      <table>
      <tr>
      <th colspan="3">Web Server Log Analyzer</th>
      </tr>
       <tr>
        <th><a href="country_wise.html">For Country Wise</a></th>
        <th> <a href="date_wise.html">For Date Wise</a></th>
        <th> <a href="os_wise.html">For Operating System Wise</a></th>
       </tr>
     </table>

</body>

</html>
```

**Output:**



Welcome to Web Server Log Analysis

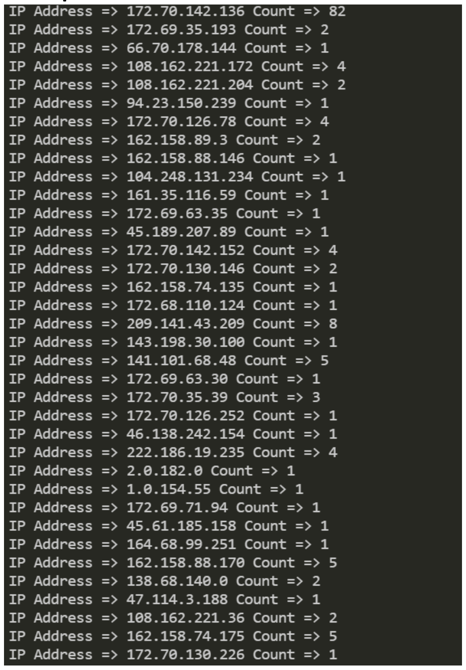
| Web Server Log Analyzer | | |
|---|---|---|
| For Country Wise | For Date Wise | For Operating System Wise |

For Country Wise:
With this we will know which country IP address just open our website or server.

Python code:

```python
import re
import csv
from collections import Counter


def access_log_reader(logfile):
    myregex = r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'

    with open(logfile) as f:
        log = f.read()
        my_iplist = re.findall(myregex, log)
        ipcount = Counter(my_iplist)
        for k, v in ipcount.items():
            print("IP Address " + "=> " + str(k) +
                  " " + "Count " + "=> " + str(v))
        with open('ipaddress.csv', 'w') as f:
            for k, v in ipcount.items():
                f.write(str(k) + "\n")


        # Create entry point of our code
if __name__ == '__main__':
    access_log_reader("access.log")
```

Output:



```
IP Address => 172.70.142.136 Count => 82
IP Address => 172.69.35.193 Count => 2
IP Address => 66.70.178.144 Count => 1
IP Address => 108.162.221.172 Count => 4
IP Address => 108.162.221.204 Count => 2
IP Address => 94.23.150.239 Count => 1
IP Address => 172.70.126.78 Count => 4
IP Address => 162.158.89.3 Count => 2
IP Address => 162.158.88.146 Count => 1
IP Address => 104.248.131.234 Count => 1
IP Address => 161.35.116.59 Count => 1
IP Address => 172.69.63.35 Count => 1
IP Address => 45.189.207.89 Count => 1
IP Address => 172.70.142.152 Count => 4
IP Address => 172.70.130.146 Count => 2
IP Address => 162.158.74.135 Count => 1
IP Address => 172.68.110.124 Count => 1
IP Address => 209.141.43.209 Count => 8
IP Address => 143.198.30.100 Count => 1
IP Address => 141.101.68.48 Count => 5
IP Address => 172.69.63.30 Count => 1
IP Address => 172.70.35.39 Count => 3
IP Address => 172.70.126.252 Count => 1
IP Address => 46.138.242.154 Count => 1
IP Address => 222.186.19.235 Count => 4
IP Address => 2.0.182.0 Count => 1
IP Address => 1.0.154.55 Count => 1
IP Address => 172.69.71.94 Count => 1
IP Address => 45.61.185.158 Count => 1
IP Address => 164.68.99.251 Count => 1
IP Address => 162.158.88.170 Count => 5
IP Address => 138.68.140.0 Count => 2
IP Address => 47.114.3.188 Count => 1
IP Address => 108.162.221.36 Count => 2
IP Address => 162.158.74.175 Count => 5
IP Address => 172.70.130.226 Count => 1
```

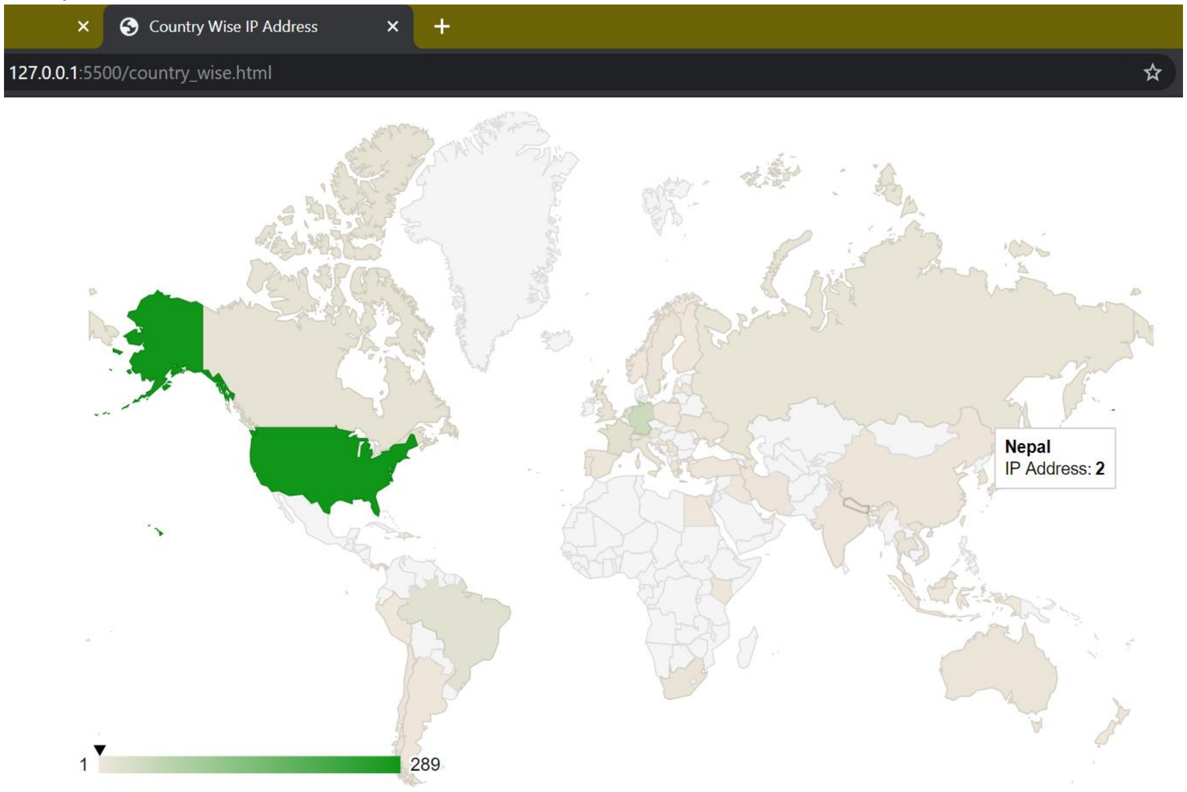
HTML/ JavaScript

Country_wise.html

```
<!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">
   <title>Country Wise IP Address</title>
</head>

<body>
   <script src="https://unpkg.com/axios@0.21.1/dist/axios.min.js"></script>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"
      integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
</script>

   <script>
      $(document).ready(function () {
         $.ajax({
            type: "GET",
            url: "ipaddress.csv",
            dataType: "text",
            success: async function (data) {
               let ipaddress = data.split("\r\n");
               let details = [];

               for (let i = 0; i < 600; i++) {
                  let resp = await fetch(`http://geoip-
db.com/json/${ipaddress[i]}`, { method: "GET" }).then(res => res.json())
                  details[i] = resp;
               }

               let newarray = details.filter((thing, index, self) =>
                  index === self.findIndex((t) => (
                     t.country_name === thing.country_name
                  ))
               )
               // console.log(details);
               // console.log(newarray);

               let final = []
               for (let i = 0; i < newarray.length; i++) {
                  let obj = ["", 0]
                  obj[0] = newarray[i].country_name;

                  let count = details.filter(i => i.country_name === obj[0]).leng
th

                  obj[1] = count;

                  final.push(obj)
               }
               // console.log(final)
               google.charts.load('current', {
                  'packages': ['geochart'],
                  'mapsApiKey': 'AIzaSyCcUq3WC6-Cidu0xjClk5DsYpsYzeS3Gk8'
               });
               google.charts.setOnLoadCallback(drawRegionsMap);

               function drawRegionsMap() {
                  var data = google.visualization.arrayToDataTable([
                     ['Country', 'IP Address'],
                     ...final
                  ]);

                  var options = {};

                  var chart = new google.visualization.GeoChart(document.getEleme
ntById('regions_div'));

                  chart.draw(data, options);
               }
            }
         });
      });
   </script>
   <!--     -->
   <script type="text/javascript">

   </script>
   <center>
      <div id="regions_div"></div>
   </center>
</body>
```

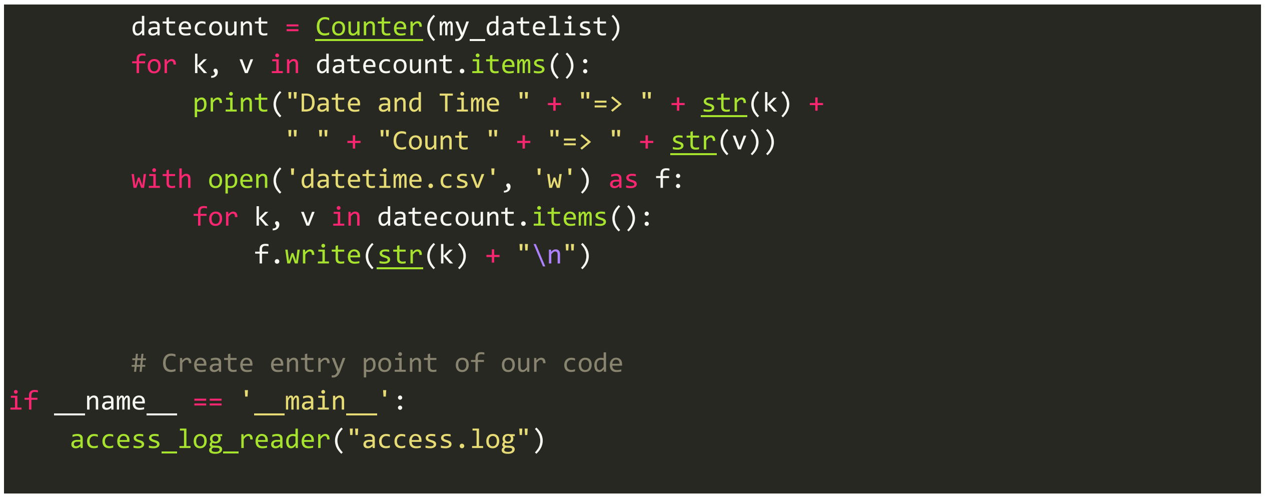
```
</html>
```

Output:



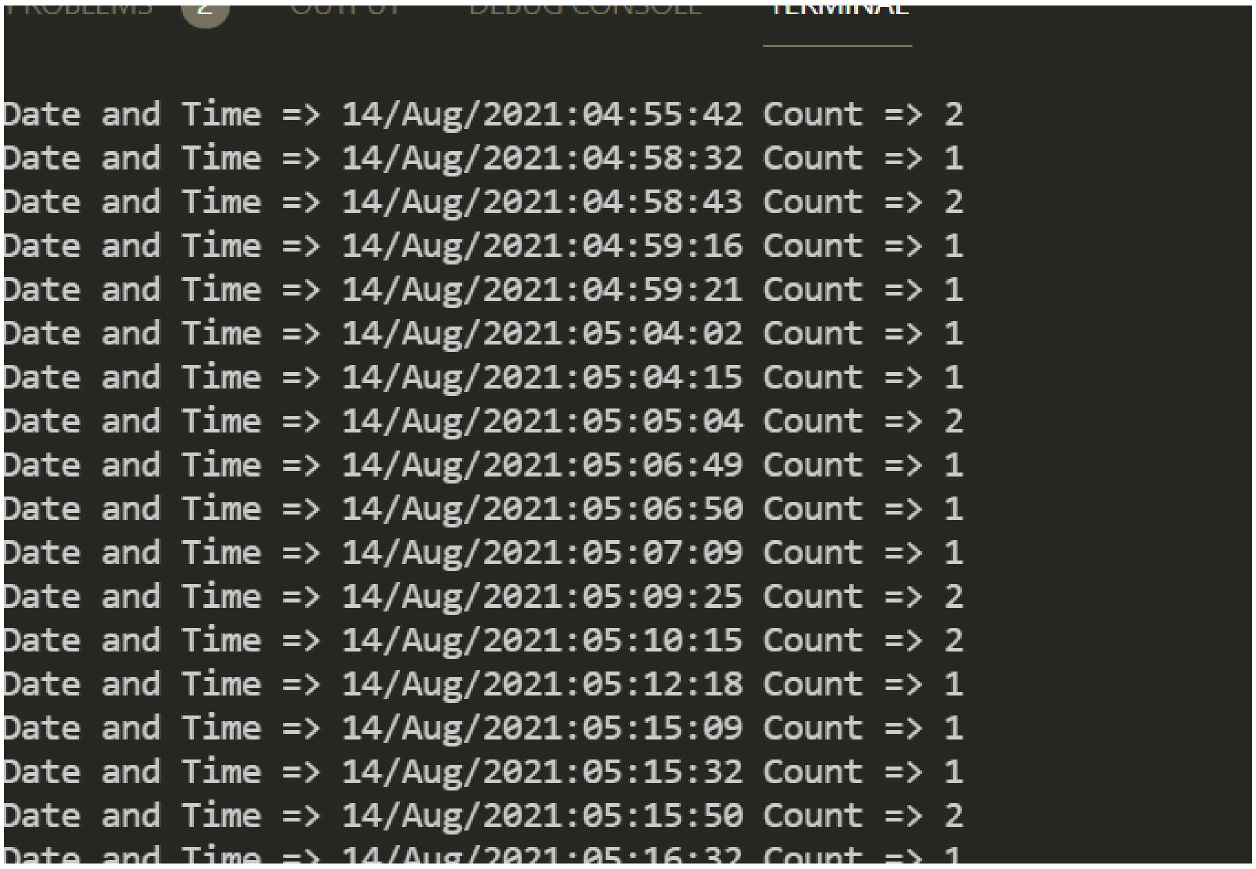
For Date wise:
It checks and record/manage the IP address according to date
Python code:
date.py

```
import re
import csv
from collections import Counter


def access_log_reader(logfile):
    myregex = r'[0-9]..\\*.[a-zA-z].\\*.[0-9].[0-9]..[0-9]..[0-9]..[0-9].'

    with open(logfile) as f:
        log = f.read()
        my_datelist = re.findall(myregex, log)
```



```
        datecount = Counter(my_datelist)
        for k, v in datecount.items():
            print("Date and Time " + "=> " + str(k) +
                  " " + "Count " + "=> " + str(v))
        with open('datetime.csv', 'w') as f:
            for k, v in datecount.items():
                f.write(str(k) + "\n")


        # Create entry point of our code
if __name__ == '__main__':
    access_log_reader("access.log")
```

Output:

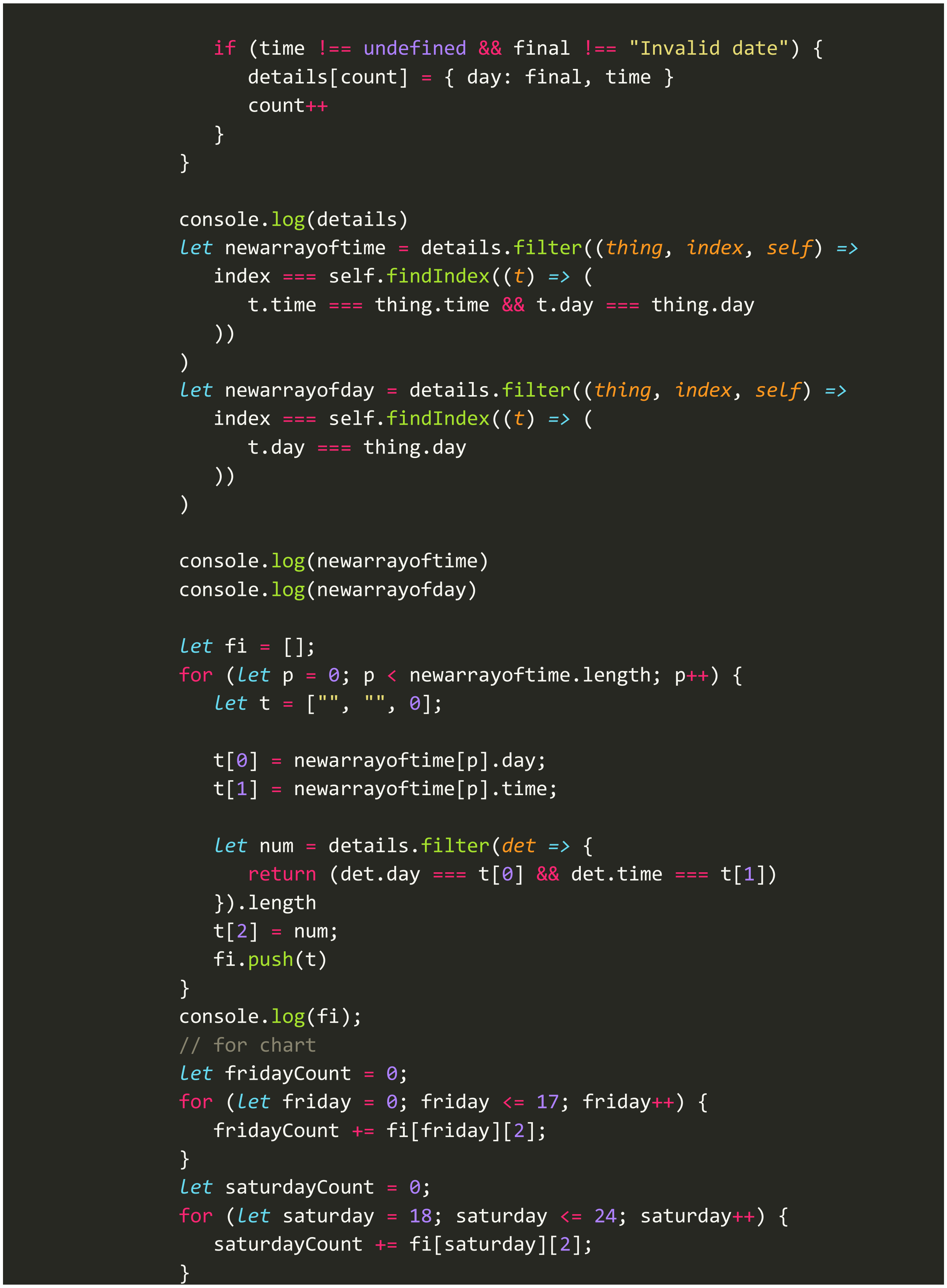


```
Date and Time => 14/Aug/2021:04:55:42 Count => 2
Date and Time => 14/Aug/2021:04:58:32 Count => 1
Date and Time => 14/Aug/2021:04:58:43 Count => 2
Date and Time => 14/Aug/2021:04:59:16 Count => 1
Date and Time => 14/Aug/2021:04:59:21 Count => 1
Date and Time => 14/Aug/2021:05:04:02 Count => 1
Date and Time => 14/Aug/2021:05:04:15 Count => 1
Date and Time => 14/Aug/2021:05:05:04 Count => 2
Date and Time => 14/Aug/2021:05:06:49 Count => 1
Date and Time => 14/Aug/2021:05:06:50 Count => 1
Date and Time => 14/Aug/2021:05:07:09 Count => 1
Date and Time => 14/Aug/2021:05:09:25 Count => 2
Date and Time => 14/Aug/2021:05:10:15 Count => 2
Date and Time => 14/Aug/2021:05:12:18 Count => 1
Date and Time => 14/Aug/2021:05:15:09 Count => 1
Date and Time => 14/Aug/2021:05:15:32 Count => 1
Date and Time => 14/Aug/2021:05:15:50 Count => 2
Date and Time => 14/Aug/2021:05:16:32 Count => 1
```

HTML/JavaScript

Date_wise.html

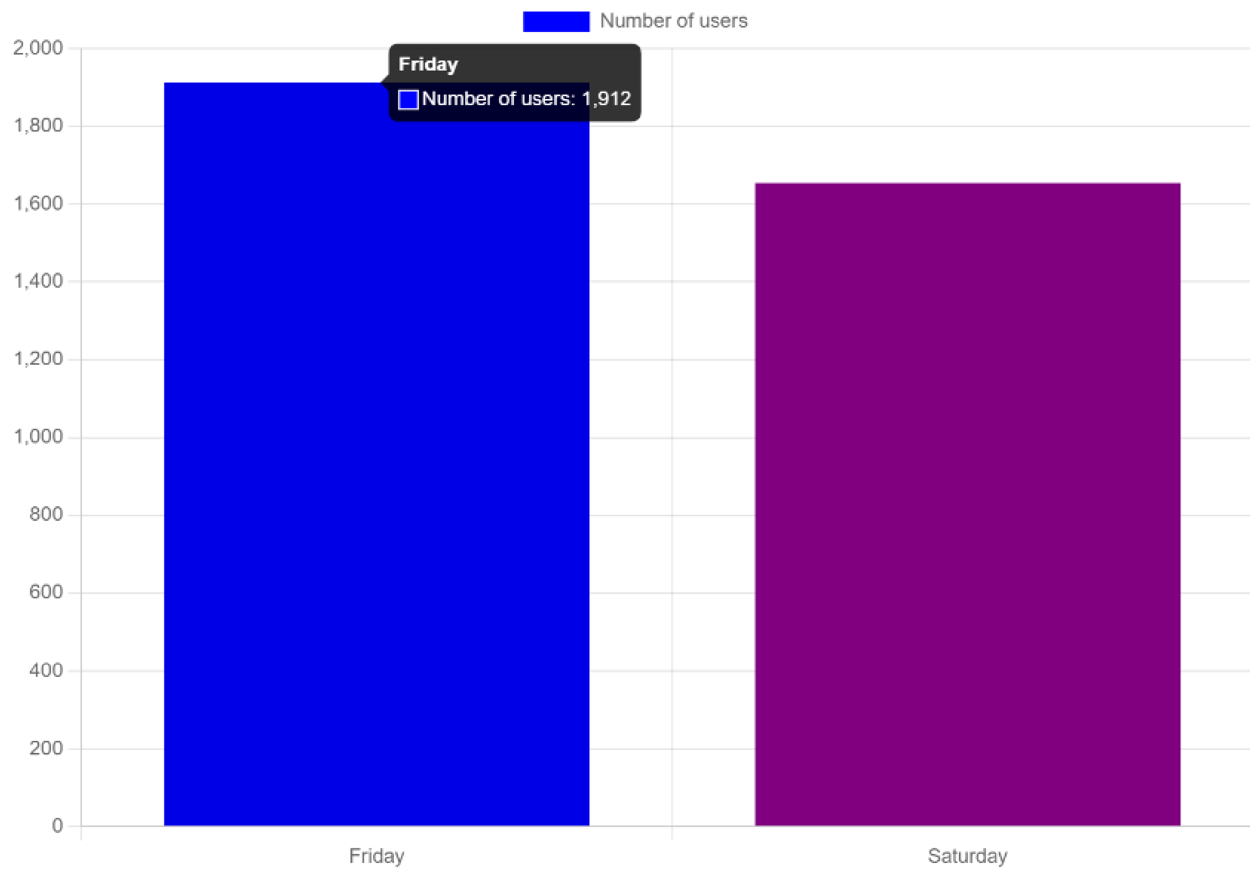
```
<!DOCTYPE html>
<html lang="en">

<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">
   <title>Time and Date - Users</title>
</head>

<body>
   <script src="https://cdn.anychart.com/releases/8.0.0/js/anychart-
base.min.js"></script>
   <script src="https://cdn.anychart.com/releases/8.0.0/themes/dark_earth.min.js"
 type="text/javascript"></script>
   <script src="https://unpkg.com/axios@0.21.1/dist/axios.min.js"></script>
   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
   <script src="https://momentjs.com/downloads/moment.min.js"></script>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"
      integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
</script>
   <script>
      $(document).ready(function () {
         $.ajax({
            type: "GET",
            url: "datetime.csv",
            dataType: "text",
            success: async function (data) {
               let datetime = data.split("\r\n");
               // console.log(datetime.length);
               // console.log(datetime);
               let details = [];
               let count = 0
               for (let i = 0; i < datetime.length; i++) {
                  let time = datetime[i].split(":");

                  let date = time[0].split("/");
                  let temp = `${date[2]}  ${getMonthFromString(date[1])}  ${date[
0]}`
                  let final = moment(temp).format("dddd");
                  time = time[1]
```



```javascript
   if (time !== undefined && final !== "Invalid date") {
      details[count] = { day: final, time }
      count++
   }
}

console.log(details)
let newarrayoftime = details.filter((thing, index, self) =>
   index === self.findIndex((t) => (
      t.time === thing.time && t.day === thing.day
   ))
)
let newarrayofday = details.filter((thing, index, self) =>
   index === self.findIndex((t) => (
      t.day === thing.day
   ))
)

console.log(newarrayoftime)
console.log(newarrayofday)

let fi = [];
for (let p = 0; p < newarrayoftime.length; p++) {
   let t = ["", "", 0];

   t[0] = newarrayoftime[p].day;
   t[1] = newarrayoftime[p].time;

   let num = details.filter(det => {
      return (det.day === t[0] && det.time === t[1])
   }).length
   t[2] = num;
   fi.push(t)
}
console.log(fi);
// for chart
let fridayCount = 0;
for (let friday = 0; friday <= 17; friday++) {
   fridayCount += fi[friday][2];
}
let saturdayCount = 0;
for (let saturday = 18; saturday <= 24; saturday++) {
   saturdayCount += fi[saturday][2];
}
```

```
               // Bar chart
               new Chart(document.getElementById("bar-chart"), {
                  type: 'bar',
                  data: {
                     labels: ["Friday", "Saturday"],
                     datasets: [
                        {
                           label: "Number of users",
                           backgroundColor: ["blue", "purple", "red", "yellow", "
green"],

                           data: [fridayCount, saturdayCount]
                        }
                     ]
                  },
                  options: {
                     legend: { display: false },
                     title: {
                        display: true,
                        text: 'Users By Date and Time'
                     }
                  }
               });
            }
         });
      });
      function getMonthFromString(mon) {
         return new Date(Date.parse(mon + " 1, 2012")).getMonth() + 1
      }
   </script>
   <div><canvas id="bar-chart" width="500" height="350"></canvas></div>

</body>

</html>
```

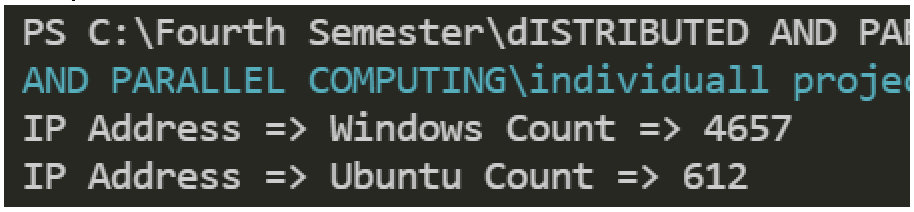
Output:



For Operating System:
This record and manage according to the user operating system
Python code:
Operatingsystem.py

```
import re
import csv
from collections import Counter


def access_log_reader(logfile):
    myregex1 = r'Windows'
    myregex2 = r'Ubuntu'
# count windows
    with open(logfile) as f:
        log = f.read()
        my_windows = re.findall(myregex1, log)
        windows_count = Counter(my_windows)
        for k, v in windows_count.items():
            print("IP Address " + "=> " + str(k) +
         " " + "Count " + "=> " + str(v))
    with open('os.csv', 'w') as f:
      for k, v in windows_count.items():
        f.write(str(k) + ","+str(v)+",")
# count ubuntu
  with open(logfile) as f:
    log = f.read()
    my_ubuntu = re.findall(myregex2, log)
    ubuntu_count = Counter(my_ubuntu)
    for k, v in ubuntu_count.items():
      print("IP Address " + "=> " + str(k) +
         " " + "Count " + "=> " + str(v))
    with open('os.csv', 'a') as f:
      for k, v in ubuntu_count.items():
        f.write(str(k) + ","+str(v))

    # Create entry point of our code
if __name__ == '__main__':
  access_log_reader("access.log")
```

Output:



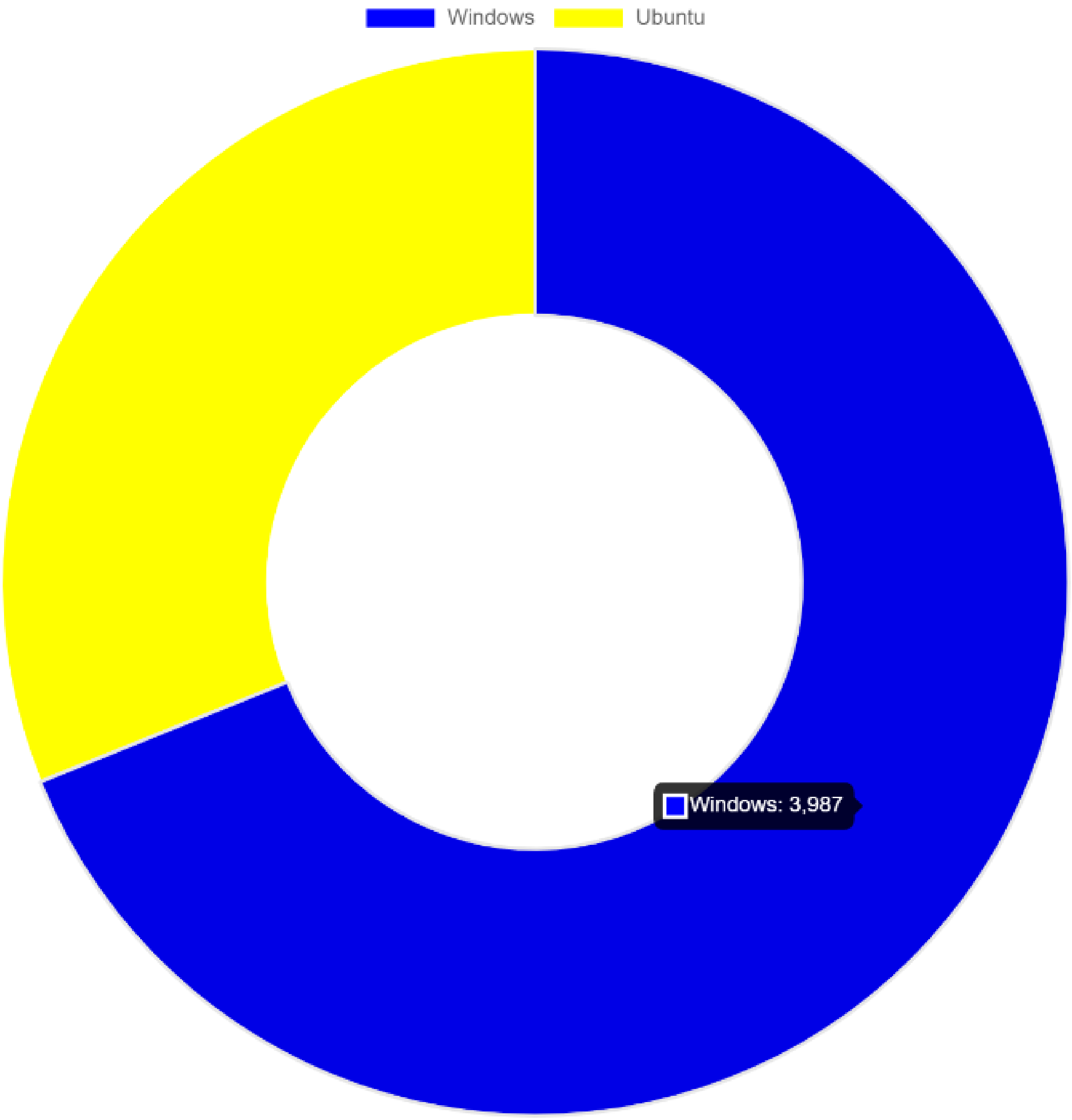
```
PS C:\Fourth Semester\dISTRIBUTED AND PA
AND PARALLEL COMPUTING\individuall proje
IP Address => Windows Count => 4657
IP Address => Ubuntu Count => 612
```

HTML/ JavaScript
Operating_System.html

```
<!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">
   <title>Operating System - Users</title>
</head>

<body>
```

```
   <script src="https://cdn.anychart.com/releases/8.0.0/js/anychart-
base.min.js"></script>
   <script src="https://cdn.anychart.com/releases/8.0.0/themes/dark_earth.min.js"
 type="text/javascript"></script>
   <script src="https://unpkg.com/axios@0.21.1/dist/axios.min.js"></script>
   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
   <script src="https://momentjs.com/downloads/moment.min.js"></script>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"
      integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
</script>
   <script>
      $(document).ready(function () {
         $.ajax({
            type: "GET",
            url: "os.csv",
            dataType: "text",
            success: async function (data) {
               let os = data.split(",");
               console.log(os.length);
               console.log(os);

               // Bar chart
               new Chart(document.getElementById("doughnut-chart"), {
                  type: 'doughnut',
                  data: {
                     labels: [os[0], os[2]],
                     datasets: [
                        {
                           label: "Number of users",
                           backgroundColor: ["blue", "yellow", "red", "green", "p
urple"],
                           data: [os[1], os[3]]
                        }
                     ]
                  },
                  options: {
                     title: {
                        display: true,
                        text: 'Users By Operating System'
                     }
                  }
               });
            }
         });
      });
   </script>
   <div class="div1">
   <canvas id="doughnut-chart" style=
"width:50%; height:10px; padding: 10px;"></canvas></div>
</body>


</html>
```

Output:



**Thank You**