**Assignment**

**CSA0814 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192324252** |
| **Name** | **K. Dedeepya** |

**Title: Log File Analyzer.**

**Problem Statement:** Develop a Python program that reads log files generated by a web server or application, extracts useful metrics such as response times, error rates, and request counts, and generates reports or alerts based on predefined thresholds.

**Code:**

**import re**

**class LogFileAnalyzer:**

**def \_init\_(self, log\_file):**

**self.log\_file = log\_file**

**self.response\_times = []**

**self.error\_count = 0**

**self.request\_count = 0**

**def analyze\_log(self):**

**with open(self.log\_file, 'r') as file:**

**for line in file:**

**self.request\_count += 1**

**response\_time = self.extract\_response\_time(line)**

**if response\_time:**

**self.response\_times.append(response\_time)**

**if self.is\_error(line):**

**self.error\_count += 1**

**def extract\_response\_time(self, log\_line):**

**# Regex to extract response time from log line, modify as per log format**

**match = re.search(r'response\_time=(\d+)', log\_line)**

**if match:**

**return int(match.group(1))**

**return None**

**def is\_error(self, log\_line):**

**# Check if log line indicates an error**

**return "error" in log\_line.lower()**

**def generate\_report(self):**

**avg\_response\_time = sum(self.response\_times) / len(self.response\_times) if self.response\_times else 0**

**error\_rate = (self.error\_count / self.request\_count) \* 100 if self.request\_count else 0**

**report = {**

**"Total Requests": self.request\_count,**

**"Average Response Time (ms)": avg\_response\_time,**

**"Total Errors": self.error\_count,**

**"Error Rate (%)": error\_rate,**

**}**

**return report**

**def check\_alerts(self, response\_time\_threshold=1000, error\_rate\_threshold=5):**

**alerts = []**

**avg\_response\_time = sum(self.response\_times) / len(self.response\_times) if self.response\_times else 0**

**error\_rate = (self.error\_count / self.request\_count) \* 100 if self.request\_count else 0**

**if avg\_response\_time > response\_time\_threshold:**

**alerts.append(f"Alert: High average response time - {avg\_response\_time}ms")**

**if error\_rate > error\_rate\_threshold:**

**alerts.append(f"Alert: High error rate - {error\_rate}%")**

**return alerts**

**if \_name\_ == "\_main\_":**

**# Replace 'server.log' with the path to your log file**

**log\_analyzer = LogFileAnalyzer('server.log')**

**log\_analyzer.analyze\_log()**

**report = log\_analyzer.generate\_report()**

**alerts = log\_analyzer.check\_alerts()**

**print("Report:")**

**for key, value in report.items():**

**print(f"{key}: {value}")**

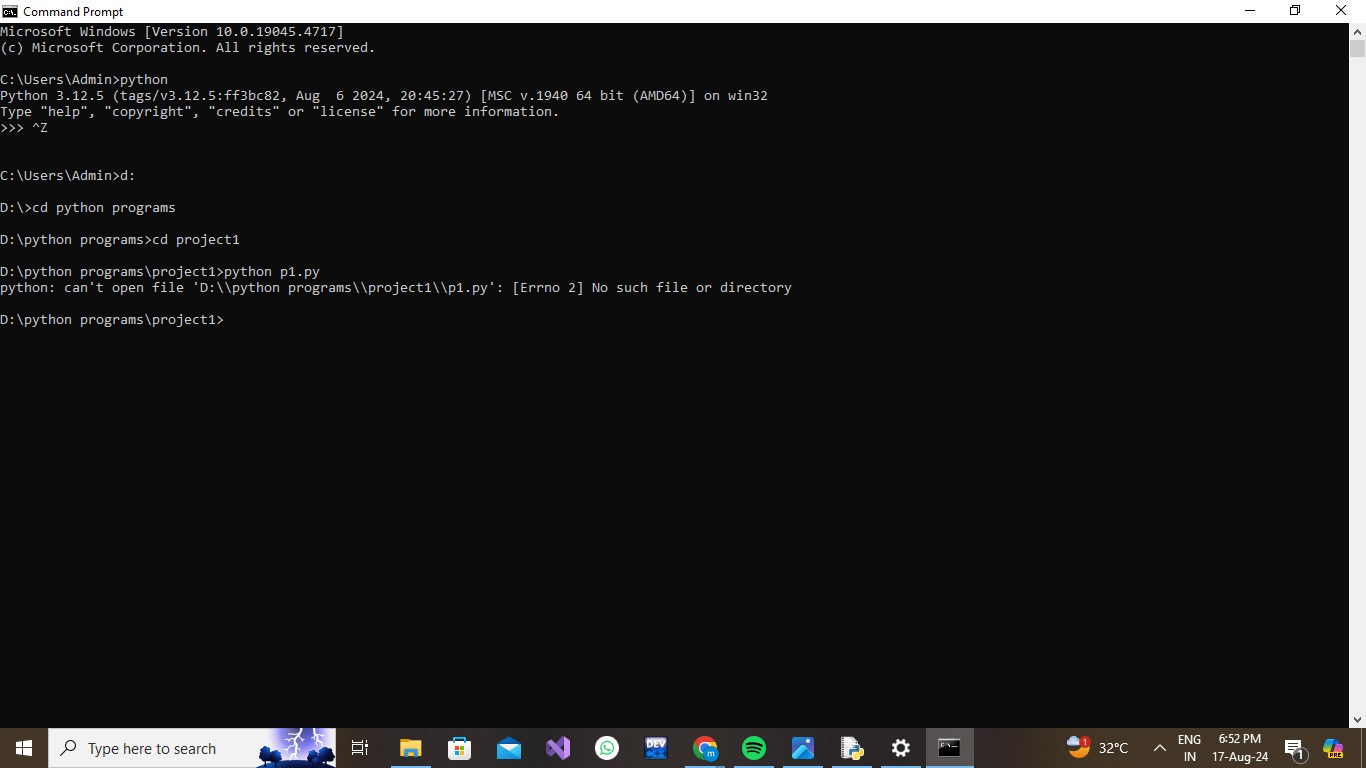
**if alerts:**

**print("\nAlerts:")**

**for alert in alerts:**

**print(alert)**

**Output Screen Shots:**



**Conclusion:**

**In conclusion, developing a Python program to analyze log files from a web server or application offers a powerful tool for monitoring and optimizing system performance. By extracting key metrics such as response times, error rates, and request counts, this program can provide valuable insights into the health and efficiency of the server. Automated reports and alerts based on predefined thresholds ensure that potential issues are identified and addressed promptly, minimizing downtime and enhancing user experience. This solution not only aids in proactive maintenance but also empowers administrators to make data-driven decisions for continuous improvement.**