README.md for `log-monitor.sh`

````
# Log Monitoring and Analysis Script

## Overview
This Bash script is designed to monitor a specified log file
in real-time and analyze its entries for predefined patterns
such as "ERROR", "DEBUG", and HTTP status codes like "404 Not
Found" and "500 Internal Server Error". It provides real-time
alerts and accumulates counts of each pattern, logging signif
icant findings for further review.

## Prerequisites
- A Linux-based system with Bash, `grep`, and `tail` installe
d.
- Read and write permissions in the directory where the scrip
t and log file are located.

## Installation
1. **Download the Script**
   - Download `log-monitor.sh` to your local system in a dire
ctory of your choiCe.

2. **Make the Script Executable**
   - Navigate to the directory where you've saved `log-monito
r.sh`.
   - Run the following command to make the script executable:
     ```bash
     chmod +x log-monitor.sh
     ```

## Usage
To start monitoring a log file, run the script with the path
to the log file as an argument:
````

```bash
./log-monitor.sh /path/to/your/logfile.log
```

Real-time Monitoring and Analysis

- The script will begin monitoring the specified log file for new entries.
- It checks each new line for the specified patterns and logs any matches found.
- The running count of each pattern's occurrences is maintained and will be displayed if the monitoring is interrupted or stopped.

Stopping the Script

- To stop the script, press `Ctrl+C`. This interruption is handled gracefully, logging the stop action and displaying the total counts of each pattern found during the session.

Log File

- All significant events and matches are logged to `log-monitor.log` in the same directory as the script. This includes each match found and a timestamp of when the monitoring was stopped.

Expected Output

Upon stopping the script with `Ctrl+C`, you'll see an output similar to the following:

```
Monitoring stopped by user.
Total occurrences of 'ERROR': 3
Total occurrences of 'DEBUG': 5
Total occurrences of '500 Internal Server Error': 2
Total occurrences of '404 Not Found': 1
```

This output summaries the counts of each pattern found during the monitoring session.

Notes

- Ensure that the log file exists and is being updated regularly for the script to function effectively.
- The script performs case-insensitive pattern matching to ensure it captures variations in log entry formats.

```
### Additional Notes
- **Customization**: Users can modify the patterns in the scr
ipt as needed to fit different log file structures or require
ments.
- **Debugging**: If no output is detected, users are encourag
ed to check the permissions and existence of the log file, an
d ensure that `log-monitor.sh` is located in a directory wher
e it has write permissions for logging.

This documentation should be included alongside your script i
n its repository or directory to assist a
```

```
import sys
import re
import logging
from tailer import follow

# Setup basic configuration for logging
logging.basicConfig(filename='log-monitor.log', level=logging.IN
                    format='%(asctime)s %(levelname)s: %(message

def monitor_and_analyze_log(file_path, patterns):
    """Function to monitor a log file and analyze it for specifi
    try:
        # Compile regex patterns to improve matching performance
        compiled_patterns = {key: re.compile(pattern) for key, p
        counts = {key: 0 for key in patterns}  # Initialize coun
```

```
        # Follow the log file for new entries
        for line in follow(open(file_path)):
            for key, pattern in compiled_patterns.items():
                if pattern.search(line):  # Check for pattern ma
                    counts[key] += 1
                    logging.info(f"Match found for {key}: {line

    except KeyboardInterrupt:
        # Handle Ctrl+C to gracefully terminate the monitoring
        logging.info("Monitoring interrupted by user.")
        print("\nMonitoring stopped by user.")
        for key, count in counts.items():
            print(f"Total occurrences of '{key}': {count}")

    except Exception as e:
        # Log any other exceptions
        logging.error(f"An error occurred: {e}")
        sys.exit(f"Error: {e}")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        logging.error("No log file specified.")
        sys.exit("Usage: python log-monitor.py <path_to_log_file

    # Define the log file path and patterns to search
    log_file_path = sys.argv[1]
    search_patterns = {
        'ERROR': r'ERROR',
        'DEBUG': r'DEBUG',
        'HTTP 404': r'404 Not Found',
        'HTTP 500': r'500 Internal Server Error'
    }
```

```
    # Start monitoring and analyzing the log
    monitor_and_analyze_log(log_file_path, search_patterns)
```

LOG

```
2024-04-22 08:30:01 INFO: User logged in successfully.
2024-04-22 08:32:15 DEBUG: Request received for user data fetch
2024-04-22 08:33:45 ERROR: Database connection failed.
2024-04-22 08:35:22 INFO: User data updated successfully.
2024-04-22 08:37:01 DEBUG: Checking system health.
2024-04-22 08:40:00 ERROR: Failed to send email notification.
2024-04-22 08:42:00 INFO: Scheduled backup completed successfull
2024-04-22 08:45:20 DEBUG: Cache cleared.
2024-04-22 08:47:33 ERROR: Error in payment processing.
2024-04-22 08:50:00 HTTP 404: Page not found.
2024-04-22 08:55:00 HTTP 500: Internal server error.
2024-04-22 09:00:00 INFO: System shutdown initiated.
2024-04-23 10:20:00 ERROR: New error logged
2024-04-23 12:00:00 ERROR: An error occurred.
2024-04-23 12:00:00 ERROR: An error occurred.
2024-04-23 12:00:00 ERROR: An error occurred.
2024-04-23 12:00:00 ERROR: An error deleted.
```