05 Log Analysis

Assignment Instructions

Objective: Build a Spark RDD application to analyze server log data, identifying and processing error entries using `foreach` for side effects, a broadcast variable for error keywords, and an accumulator to track error counts.

Requirements:

1. **Input**: Use either the provided sample log data (in the code) or a text file (`logs.txt`) with log entries in the format: `timestamp method endpoint status message` (e.g., "2023-10-01 10:00:00 GET /home 200 OK").
2. **Broadcast Variable**: Create a broadcast variable containing a list of error codes (e.g., ["404", "500"]). Use it to check if a log entry contains an error.
3. **Accumulator**: Initialize an accumulator to count the total number of log entries containing error codes.
4. **foreach for Side Effects**: Use `foreach` to process each log line. For lines with errors, write the line to an output file (`error_logs.txt`) with a custom prefix (e.g., "[ERROR DETECTED]") or print to console if file writing is unavailable.
5. **Output**: After processing, print the total error count from the accumulator. Ensure the output file (if used) contains all error lines with proper formatting.
6. **Environment**: Use PySpark. Test locally or on a provided Spark cluster. Ensure no local file I/O for the input if running in a restricted environment (use sample data instead).

Tasks:

1. Complete the `process_log_line` function to:
   - Access error keywords via the broadcast variable.
   - Check if the log line contains any error code.
   - Increment the accumulator for each error found.
   - Write error lines to `error_logs.txt` (or print) with a prefix like "[ERROR DETECTED]".
2. Implement the `main` function to:
   - Initialize the broadcast variable and accumulator.
   - Apply `foreach` to process the RDD.
   - Print the final error count.
3. Test with the sample data and, if possible, a larger log file.

Deliverables:

- Submit the completed `log_analysis_assignment.py` file.
- Include a brief report (text or comments) explaining how you used `foreach`, the broadcast variable, and the accumulator.
- If using a file, provide a sample `logs.txt` with at least 10 entries.

Evaluation Criteria:

- Correct use of `foreach` for side effects (e.g., file writing or printing).
- Proper implementation of broadcast variable and accumulator.
- Accurate error detection and counting.
- Clean, commented code with clear logic.
- Successful execution with provided or custom input.

Sample Output:

- Console: `Total errors found: 2`
- `error_logs.txt`:

```
[ERROR DETECTED] 2023-10-01 10:01:00 GET /api 404 Not Found
[ERROR DETECTED] 2023-10-01 10:02:00 POST /login 500 Internal Server Error
```

Hints:

- Use `sc.broadcast(error_keywords)` for the broadcast variable.
- Use `sc.accumulator(0)` for the error counter.
- For file writing in `foreach`, consider using Python's `open` in append mode, but ensure thread safety or test with console output if restricted.
- Test with varied inputs (e.g., logs with no errors, all errors, or mixed).