1. Explain your decision to use dependency injection in this project. What benefits does it provide?
2. Why did you implement IParser as an interface rather than just creating the parser class directly?
3. What advantages does using a record type for LogEntry provide over a regular class?
4. Explain how the **yield return** mechanism works in your parser and why it's beneficial for large log files.
5. Walk me through your LINQ query in the TopMessages method. How does each part contribute to finding the most frequent messages?
6. How would you modify your parser to handle a different log format without changing the rest of the application?
7. What's the time complexity of your CountByType and TopMessages methods?
8. Explain your approach to handling cross-platform line endings (**&#x0a;** vs **\n**). Why was this necessary?
9. Why did you choose to make LogAnalyzer and IParser singletons in the dependency injection container?
10. How would you unit test the LogParser class? What edge cases would you consider?
11. If you needed to parse logs in real-time as they're being written, how would you modify your application?
12. Explain the C# range operator syntax (**rest[..firstSpace]**) you're using in the parser. What does it do?
13. How does your error handling strategy work? What happens if a malformed log entry is encountered?
14. What would you change or improve in your implementation if you had more time?
15. How would you scale this application to handle log files with millions of entries?
16. Explain why you call **StateHasChanged()** in some methods but not in others. How does Blazor's rendering cycle work?
17. How would you add filtering functionality to search for specific messages or filter by date ranges?
18. Explain how tuple types are used in your application and what benefits they provide.
19. If you needed to export the analysis results to a CSV or Excel file, how would you implement that?
20. How would you extend this application to visualize trends over time using the timestamp data you're already collecting?