**Performance Analysis Results**

As part of the performance optimization phase of the project, I conducted a detailed analysis of several query patterns using MongoDB’s .explain() method through PyMongo. This analysis was crucial in identifying inefficiencies in query execution and determining where indexes could significantly improve database performance.

Initially, many of the core queries relied heavily on full collection scans (COLLSCAN). This included commonly executed queries such as fetching users by their email address, retrieving assignments based on due dates, and searching for courses by title and category. The .explain() output confirmed that these queries were not taking advantage of any existing indexes, resulting in MongoDB scanning every document in the collection to return results. This is acceptable for small datasets but becomes a major performance bottleneck as the database grows.

For example, when searching for a user by their email, the .explain() method reported a COLLSCAN, indicating no index was used. To resolve this, I created a unique index on the email field in the users collection. Once the index was in place, re-running the .explain() showed that the query plan now used an IXSCAN (index scan), confirming that the database engine was leveraging the new index to return results more efficiently.

A similar issue occurred with the assignments collection. Queries that filtered assignments based on upcoming due dates were initially performing full collection scans. After implementing a single-field index on the due date field, these queries began using IXSCAN instead of COLLSCAN, resulting in faster execution and lower resource usage.

Another optimization was made for the courses collection, where queries were filtering documents based on both title and category. The original queries again triggered a COLLSCAN, which was inefficient. I introduced a compound index on the title and category fields. This optimization enabled the database to narrow down results using a single, efficient index traversal rather than scanning the entire dataset.

To validate the impact of these optimizations, I used Python’s time module to measure query performance before and after the indexes were applied. In every case, the optimized queries showed significant reductions in execution time sometimes improving from over 200 milliseconds down to less than 10 milliseconds.

The use of .explain() proved essential not only in diagnosing performance problems but also in confirming the benefits of each optimization. The shift from COLLSCAN to IXSCAN across multiple critical queries demonstrated how indexing, when aligned with access patterns, leads to measurable improvements in speed, efficiency, and scalability. This process reinforced the importance of proactive query analysis and index planning in designing high-performance mongodb applications.