Jean Pierre Jacques Toussaint

Professor Sandifer

Southern New Hampshire University

CS-499-19008

November 20, 2025

The artifact I selected for this enhancement is the backend portion of the Travlr Getaway application, initially developed in CS-465 Full-Stack Development. The system was built using the MEAN stack (MongoDB, Express, Angular, and Node.js) and serves as a travel booking and trip management platform. The artifact includes the trip routes, controllers, models, and associated logic for retrieving and managing trip data. The original version of the artifact was created as part of the course's final project, with the primary goal of building a functional full-stack application with CRUD operations and basic authentication.

I selected this artifact because it provides a meaningful opportunity to demonstrate algorithmic thinking, performance optimization, and the use of data structures within a real, production-style application. The existing trip-retrieval logic relied on simple list queries using skip() and limit(), which became inefficient on large datasets. This made the artifact an ideal candidate for algorithmic enhancement.

For this milestone, I implemented cursor-based pagination, indexed filtering, and range-based search algorithms, which significantly improve performance and scalability. Cursor-based pagination relies on using MongoDB's _id field as a natural ordering mechanism, eliminating the need for the database to scan and skip large numbers of documents. I also added dynamic

filtering logic based on resort name and date ranges, leveraging MongoDB indexes to improve query precision.

These enhancements demonstrate the following algorithmic and data-structure skills:

- Designing and implementing efficient search algorithms
- Improving time complexity by removing skip-based iteration
- Using MongoDB's indexed object IDs as cursor markers
- Applying compound and field-based indexing
- Structuring API logic to operate efficiently at scale

The optimized search route now supports:

- limit parameters
- Encoded cursors (nextCursor) for efficient forward traversal
- Filtering by resort substring
- Date range queries using comparison operators
- Graceful handling of invalid cursor tokens

These improvements convert a basic list-retrieval function into a scalable, algorithmic search system better aligned with industry expectations for API design.

Yes, the enhancement meets the course outcomes I targeted in Module One, particularly CO3, which focuses on designing and evaluating computing solutions using algorithmic

principles and appropriate data-structure techniques. Implementing cursor-based pagination required analyzing performance trade-offs, understanding how MongoDB organizes and indexes data, and structuring queries to optimize time complexity.

This enhancement also supports CO4, which emphasizes the use of innovative, well-founded techniques to implement computing solutions. The implementation required thoughtful design, input validation, indexing strategies, and the construction of an efficient search algorithm within a real-world backend.

At this point, no significant updates to outcome coverage are needed.

The enhancement successfully demonstrates progress toward CO3 and supports the overall ePortfolio theme of performance, scalability, and modern backend engineering.

Enhancing this artifact taught me how algorithmic decisions directly affect performance, maintainability, and scalability in backend architecture. I learned how to replace offset-based pagination with cursor-based pagination, how to use indexed fields effectively, and how to encode and decode cursor tokens securely. This process improved my understanding of time-complexity trade-offs and helped me recognize why large-scale systems rarely rely on skip-based pagination.

I also learned how to design a search API that remains predictable for the client while still being efficient internally. This required thoughtful decisions about response structure (trips, nextCursor), validation logic, and handling edge cases such as invalid cursors or empty result sets.

Some challenges included:

- Decoding and validating cursor tokens
- Ensuring compatibility with existing trip structures and front-end expectations
- Avoiding breaking changes while adding new functionality
- Testing multiple combinations of filter, date, cursor, and limit parameters
- Ensuring returned queries used appropriate indexes

These challenges strengthened my debugging and problem-solving abilities and pushed me to reason about data flow, query design, and efficient iteration.

Overall, this milestone helped deepen my understanding of algorithms in a practical context. Instead of writing standalone algorithmic exercises, I applied algorithmic thinking to a realistic API used by an end-to-end application, mirroring real engineering work.