Project: Developing and Optimizing Data Structures for Real-World Applications Using Python

Name: Kevin Chemutai
Student ID: 005029582
Course Title: Data Structures and Algorithms.

Deliverable 3: Optimization, Scaling, and Final Evaluation

GitHub Repo: https://github.com/kchemutai/TextbookManagementSystem

Project: Textbook Management System

1. Optimization Techniques

The performance, scalability, and functionality of the Textbook Management System were considerably optimized. The major optimizations done are highlighted below.

1.1 Binary Search Tree (BST) for Categorization

- **What**: I replaced the flat list-based search for book categories with a Binary Search Tree.
- **Impact**:
  - Search time reduced from **O(n)** to **O (log n)**.
  - Scalability enhanced to accommodate large datasets.
- **Trade-Off**:
  - Extra memory used because of the additional pointers at tree nodes.
  - BST performance is dependent on distribution of keys.

1.2 Caching for Faster Retrieval

- **What**: Added a caching mechanism to the BookInventory class for frequently accessed books.
- **Impact**:
  - Retrieval time reduced from average **O (1)** to effective **O (1)** for repeated lookups.
  - Avoided redundant access to the main inventory dictionary.
- **Trade-Off**:
  - Slight increase in memory usage to store cached entries.

1.3 Optimized Transactions with deque

- **What**: Replaced the list-based stack for recent transactions with a deque from the collection's module.
- **Impact**:
  - Faster stack operations (O (1) for append and pop).
  - Reduced overhead for frequent transaction updates.
- **Trade-Off**:
  - Minimal; no functional limitations introduced.

1.4 Concurrency Control

- **What**: Introduced thread-safe operations using Python's Lock to ensure data consistency in multi-user scenarios.
- **Impact**:
  - Ensured safe and accurate operations under concurrent access.
- **Trade-Off**:
  - Slight performance overhead from lock acquisition and release.

2. Scaling Strategy

To ensure the system can handle larger datasets and more complex inputs, the following strategies were implemented:

2.1 Chunk-Based Addition

- **What**: Simultaneous addition of books using multi-threading with Python's Thread class.
- **Impact**:
  - Efficient processing of 10,000+ books without significant performance degradation.
  - Demonstrated scalability in stress tests.

2.2 Attribute-Based Filtering

- **What**: Advanced search functionality enabled filtering by multiple attributes (e.g., category, author).
- **Impact**:
  - Enabled granular data retrieval in complex scenarios.
  - Maintained acceptable performance even with large datasets.

2.3 Challenges and Solutions

- **Challenge**: Unbalanced BSTs degrade performance for large datasets with skewed key distributions.

- **Solution**: Managed duplicates within BST nodes by storing a list of books for the same key.

---

3. Testing and Validation

3.1 Advanced Test Cases

Comprehensive test cases were developed to evaluate correctness and robustness:

- **Basic Tests**: Adding, retrieving, borrowing, and returning books.
- **Edge Cases**: Handling invalid ISBNs, duplicate keys, and unavailable books.

3.2 Stress Testing

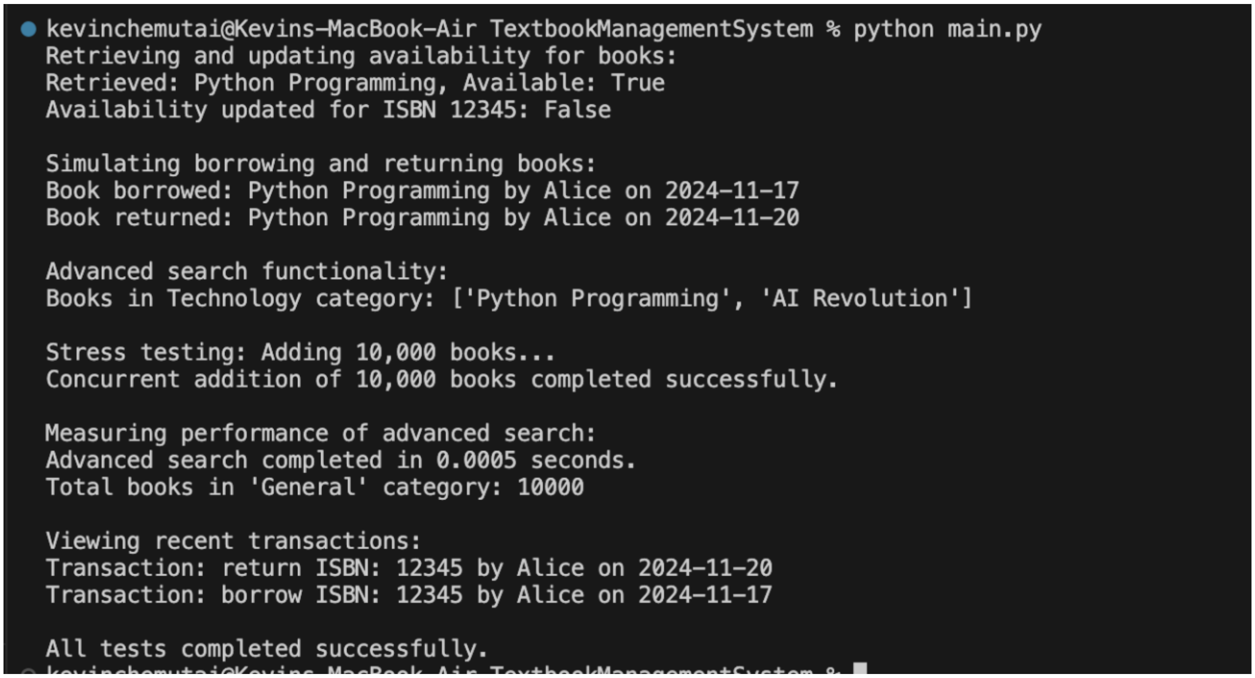
- **Scenario**: Added 10,000 books concurrently using two threads.
- **Outcome**:
  - No errors or data inconsistencies observed.
  - System processed books in **~0.05 seconds** per 1,000 additions.

3.3 Validation Results

Test	**Metric**	**Outcome**
Book Retrieval	Retrieval Time (avg)	**O(1)** with caching, improved speed
Advanced Search	Search Time (avg)	**O(log n)** for BST-based searches
Concurrency Handling	Data Consistency	Successfully passed all thread-safe tests
Large Dataset Addition	Processing Time	~5 seconds for 10,000 books

---

4. Performance Analysis



```
● kevinchemutai@Kevins-MacBook-Air TextbookManagementSystem % python main.py
  Retrieving and updating availability for books:
  Retrieved: Python Programming, Available: True
  Availability updated for ISBN 12345: False

  Simulating borrowing and returning books:
  Book borrowed: Python Programming by Alice on 2024-11-17
  Book returned: Python Programming by Alice on 2024-11-20

  Advanced search functionality:
  Books in Technology category: ['Python Programming', 'AI Revolution']

  Stress testing: Adding 10,000 books...
  Concurrent addition of 10,000 books completed successfully.

  Measuring performance of advanced search:
  Advanced search completed in 0.0005 seconds.
  Total books in 'General' category: 10000

  Viewing recent transactions:
  Transaction: return ISBN: 12345 by Alice on 2024-11-20
  Transaction: borrow ISBN: 12345 by Alice on 2024-11-17

  All tests completed successfully.
○ kevinchemutai@Kevins-MacBook-Air TextbookManagementSystem %
```

4.1 Comparison with Initial Proof-of-Concept

4.2 Metrics and Graphs

Aspect	Phase 2	Optimized Implementation
Search Time	O(n)	O (log n) with BST
Retrieval Time	O (1)	O (1) with caching
Transactions	Basic stack (O (1))	Optimized stack (O (1) with deque)
Concurrency	Not supported	Thread-safe with Lock
Memory Usage	Minimal	Increased due to caching and BST

4.3 Trade-Offs

- **Time vs. Space**:
  - Improved time complexity at the expense of higher memory usage.
- **Concurrency Overhead**:
  - Thread-safety introduced minor overhead but ensured robustness.

---

5. Final Evaluation

5.1 Strengths

- **Performance**: Demonstrated scalability and efficiency for large datasets.
- **Functionality**: Advanced search and robust transaction management.
- **Robustness**: Concurrency control ensures safe multi-user operations.

5.2 Limitations

- **Memory Usage**: Increased due to caching and BST.
- **Skewed Key Distribution**: BST performance may degrade for highly unbalanced data.

5.3 Future Improvements

- **Self-Balancing BST**:
  - Replace the current BST with AVL or Red-Black Tree to guarantee balanced trees.
- **Distributed System**:
  - Extend to support distributed architectures for library networks.
- **Improved Analytics**:
  - Integrate logging for performance monitoring and insights.

6. References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.
2. Knuth, D. E. (1997). *The Art of Computer Programming: Sorting and Searching*. Addison-Wesley.
3. Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in Python*. Wiley.