Online Marketplace: Inventory & Order Matching Engine

Week 12: Core Feature Demo

- Course: Data Structures & Algorithms
- Team Members: Bilal Azfar, Huzaifa Soomar, Musab Abbasi

From Design to Working Implementation

Since our last review in Week 10, significant strides have been made, transforming our conceptual designs into a tangible, testable system.

Core Data Structures in C++

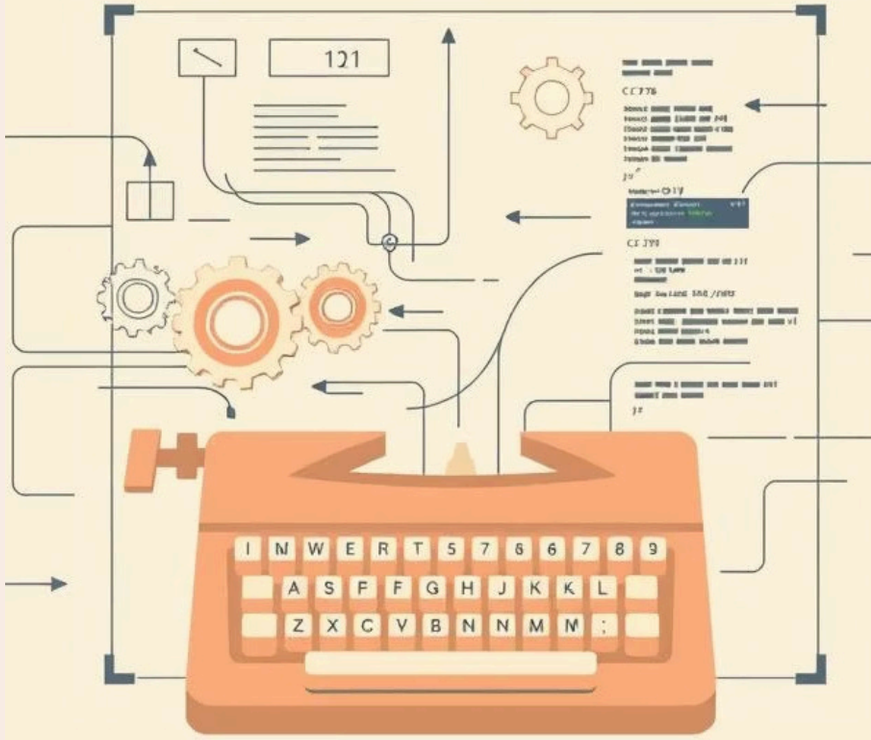
The foundational data structures are now robustly implemented using C++, ensuring efficiency and performance.

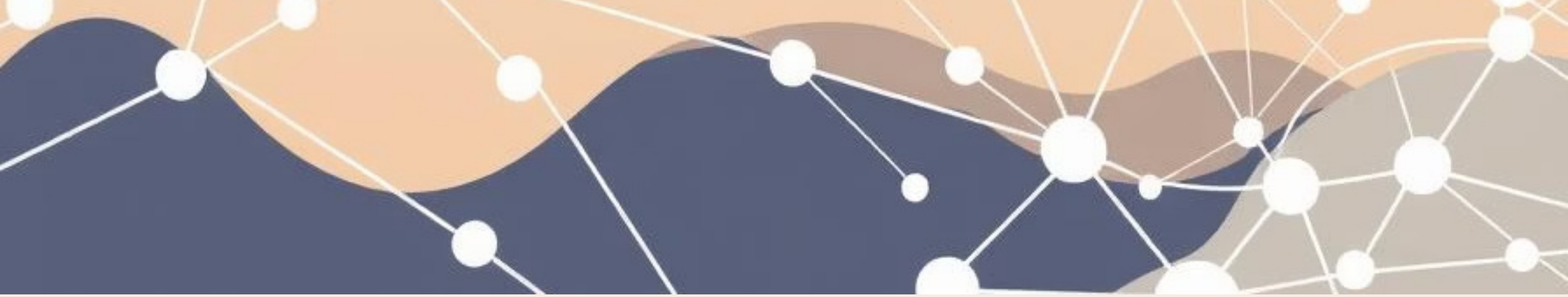
Executable & Testable Modules

Individual modules are now fully functional, allowing for isolated testing and integration validation.

Live Demo Prepared

A comprehensive live demonstration has been prepared, utilizing carefully selected test cases to showcase functionality.





Implemented Modules (Week 12 Scope)

For this Week 12 demonstration, we are focusing on the modules that have reached a fully implemented and tested state.

Catalog: HashMap

Efficient product storage and retrieval.

Category Hierarchy: N-ary Tree

Organized categorization for easy navigation.

OrderBook: Max-Heap / Min-Heap

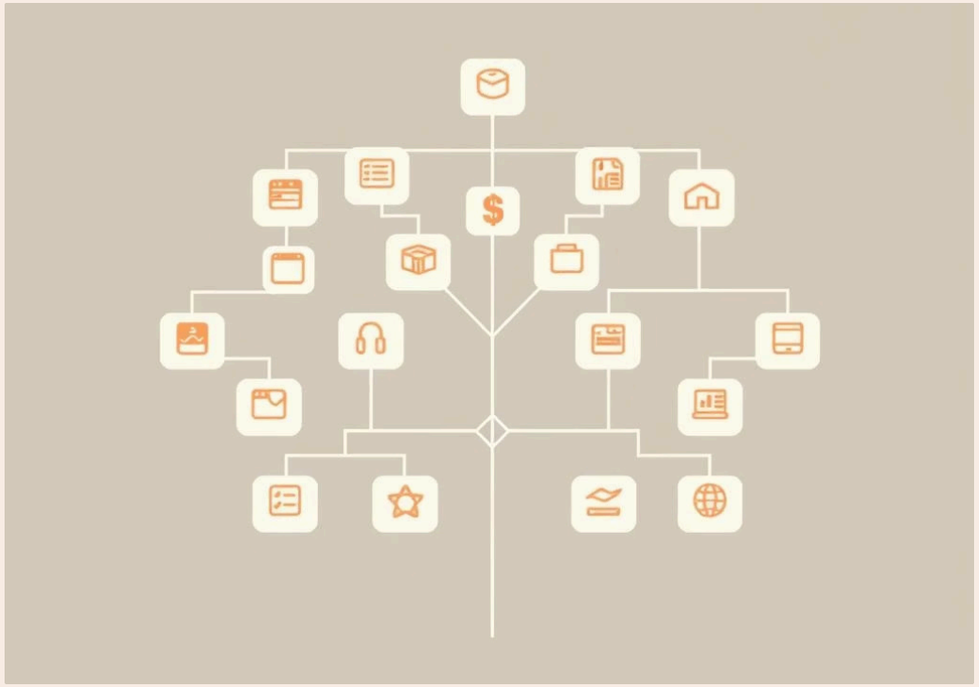
Prioritized management of buy and sell orders.

Recommendation Engine: Graph

Generates personalized product suggestions.

Note: This scope focuses on implemented features, not the full matching engine, as correctly specified.

Category & Catalog Demonstration



Data Structures Used

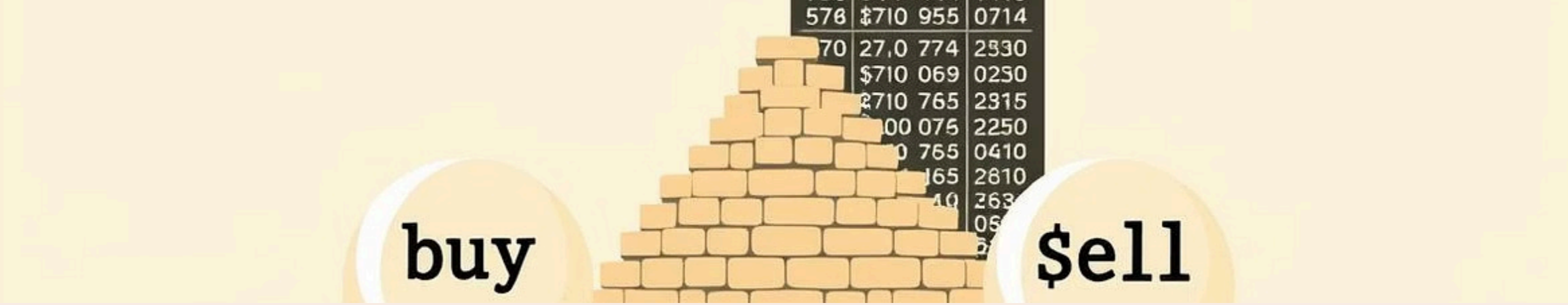
- Category Tree: **N-ary Tree**
- Product Lookup: **HashMap**

Demo Highlights

- Efficient category insertion and hierarchy management.
- Rapid category search for streamlined user experience.
- O(1) average time complexity for product lookup.

This demonstration showcases the backbone of our inventory management, allowing for intuitive organization and lightning-fast retrieval of products.

The N-ary tree structure enables flexible category management, while the HashMap guarantees optimal performance for direct product access.



OrderBook & Priority Handling

What is Demonstrated

- **Buy Orders:** Managed efficiently using a **Max-Heap**, prioritizing highest bids.
- **Sell Orders:** Handled with a **Min-Heap**, prioritizing lowest asks.
- **Best Order Retrieval:** Demonstrates quick access to the most favorable prices by time and value.

The OrderBook module is crucial for maintaining market efficiency. By using specialized heap structures, we ensure that the most competitive buy and sell orders are always immediately accessible. This is a critical prerequisite for advanced matching functionalities.

> This stage primarily demonstrates the robust order prioritization mechanism, laying the groundwork for the full matching engine.

Graph-Based Recommendation Engine

Our recommendation engine leverages graph theory to provide intelligent and personalized product suggestions.

Graph Nodes: Products

Each unique product in our marketplace is represented as a node within the graph.

Edge Weight: Co-purchase Frequency

The strength of the connection between products is determined by how often they are purchased together.

Top-K Recommendations

The engine efficiently identifies and returns the top 'K' most relevant recommendations.

Demo Highlights

The demo dynamically showcases the system's ability to record co-purchase events and generate sorted recommendations in real-time, enhancing the user shopping experience.

Demo Test Cases: Ensuring Reliability

To guarantee a smooth and reliable demonstration, we have prepared a suite of hardcoded test cases, avoiding any external dependencies during the live presentation.

Category Operations

Tests for successful insertion, deletion, and search within the category hierarchy.

Product Lookup

Verification of instant product retrieval using the HashMap.

OrderBook Mechanics

Confirmation of correct best buy/sell order retrieval from heaps.

Recommendation Output

Validation of accurate and sorted product recommendations.

Note: No file I/O will be performed during the live demonstration to maintain focus and control.

Performance Expectations (Big-O Analysis)

Our design prioritizes efficiency, and the Big-O notation reflects the expected performance characteristics of our core operations.

O(1)

HashMap Search

Average time complexity for product lookup.

O(log n)

Heap Insert/Remove

Efficient order book operations.

O(n)

Category Tree Search

Worst-case scenario for deep category searches.

O(k log k)

Recommendation Sorting

For returning top 'k' sorted recommendations.

While no benchmarking has been performed yet, these theoretical bounds guide our architectural choices and ensure scalability.

Features Not Yet Implemented

Transparency is key to project management. We openly acknowledge the features that are beyond the scope of this Week 12 implementation.

Core Matching Logic

- Full matching loop execution
- Partial order fills
- Trade execution protocols

Ancillary Modules

- Inventory updates post-match
- Data persistence (files/DB integration)
- Comprehensive reporting module for inventory analytics

These elements represent the next phases of development and will be addressed in subsequent iterations.

Next Steps & Conclusion

Next Steps

- Implement the full **MatchingEngine** module.
- Integrate robust order execution mechanics.
- Expand recommendation logic for greater accuracy.
- Develop a comprehensive inventory reporting module.
- Explore optional file-based input for larger datasets.

Conclusion

This Week 12 demonstration successfully validates the implementation of our core features. We have effectively applied complex data structures to build functional modules for our online marketplace.

The team has met the requirements for this phase, setting a strong foundation for the subsequent development of the full matching engine.