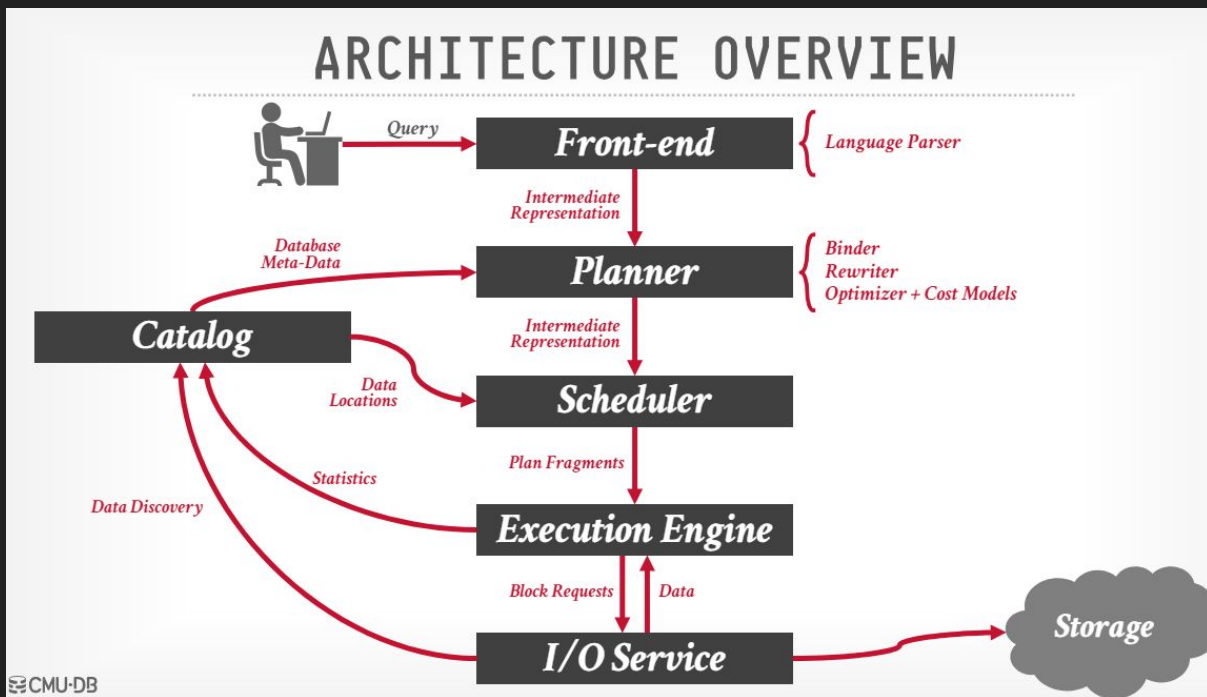


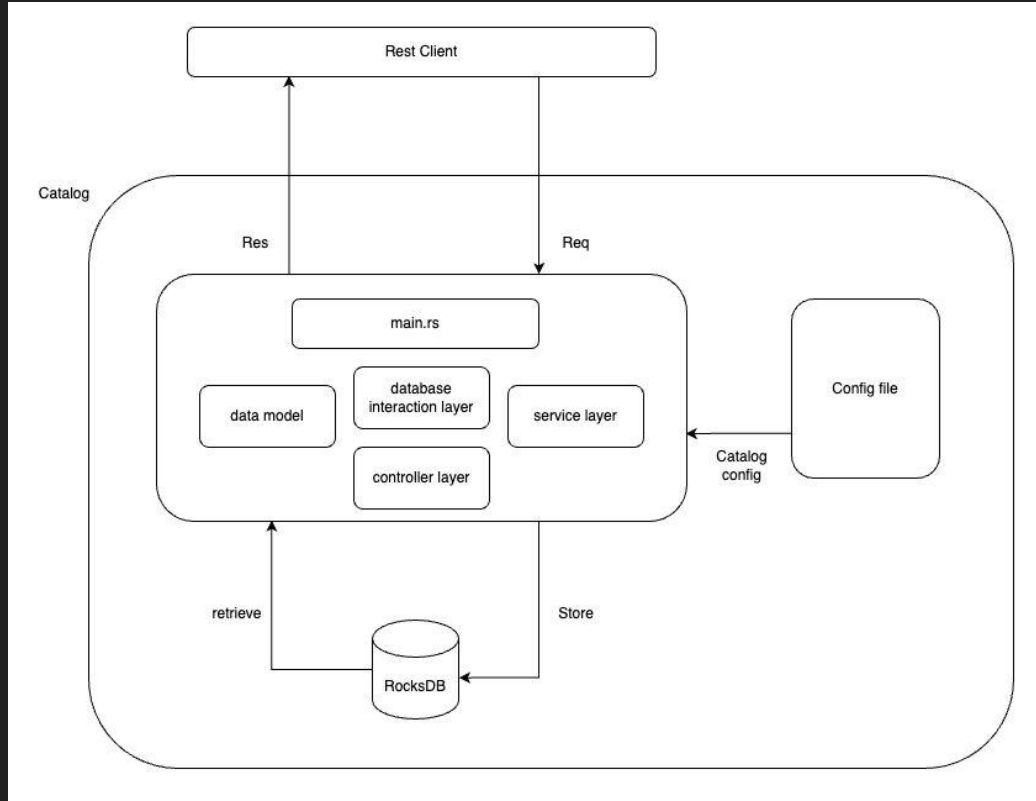
Catalog Service Team - 1

Final Project Update

The Catalog Service

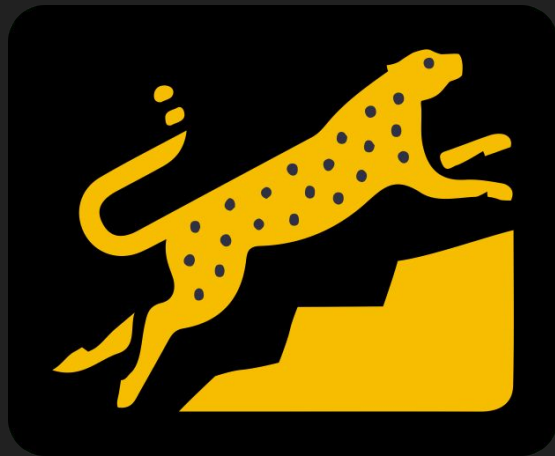


Architecture



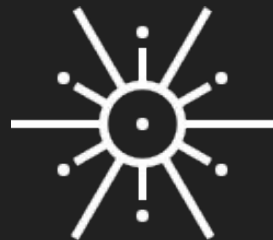
Using RocksDB

- Concurrency Control
- Flexibility
- Scalability
- Supports high-frequency reads



Using Axum

- Asynchronous Capabilities
- Scalability and Minimal Overhead
- Build on top of Tokio and Hyper



Testing

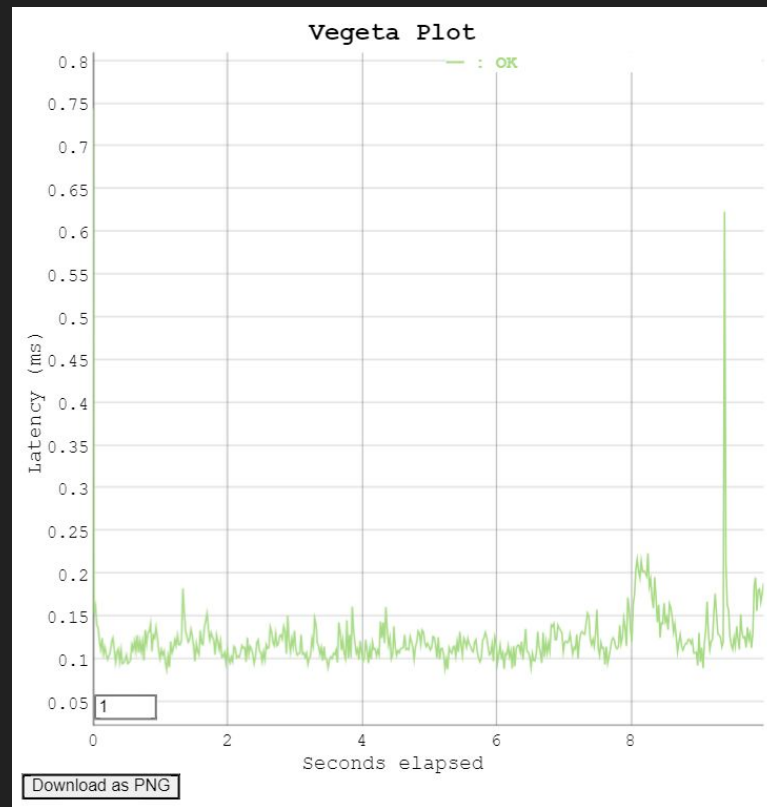
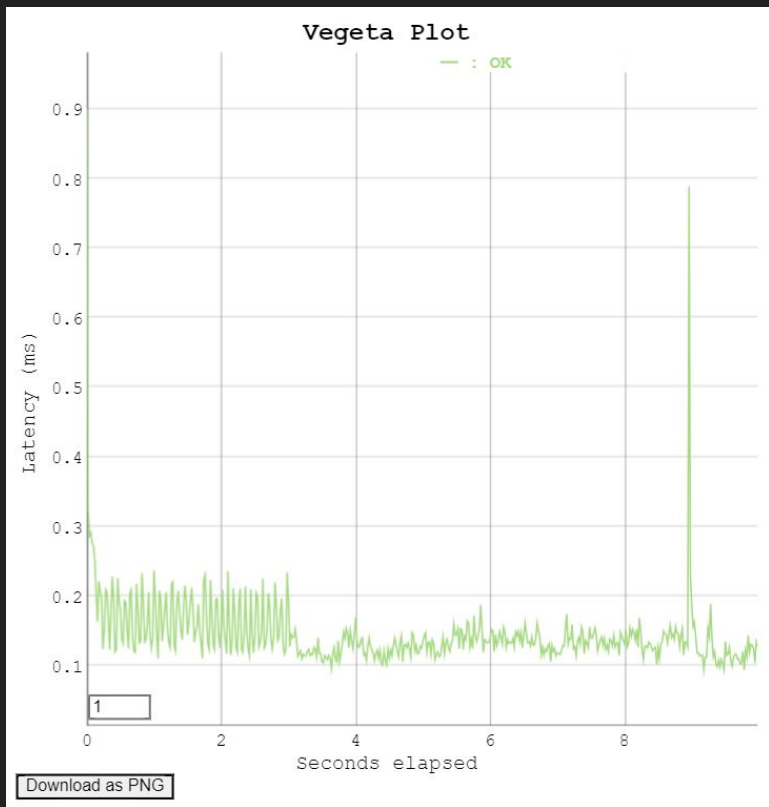
- Unit test coverage: 84% ✓
- Performance Testing
 - Support for multiple clients ✓
 - Mixed Workloads ✓
 - Large Schema or Datasets ✕

Performance Testing Results




Latency and Throughput

Operation	Mean Latency	Throughput
Get namespace	182.686 μ s	50.10
Get table	159.402 μ s	50.10
List Namespace	188.989 μ s	50.10
List Tables	267.312 μ s	50.10
Random requests	129.312 μ s	50.02

Latency - Get Namespace



Milestones

-  75%: Basic API support
-  100%: Support for concurrent requests
- 125% (targetted) : Performance test against Iceberg
Catalog
-  Actual completion: Performance testing
(Thanks to Catalog Team 2 for the benchmarking script)

Future Work

- Support for true parallelism using thread pool
- Performance tuning of RocksDB
- Add Regression Tests
- Test with large Schema or Datasets

Thank you