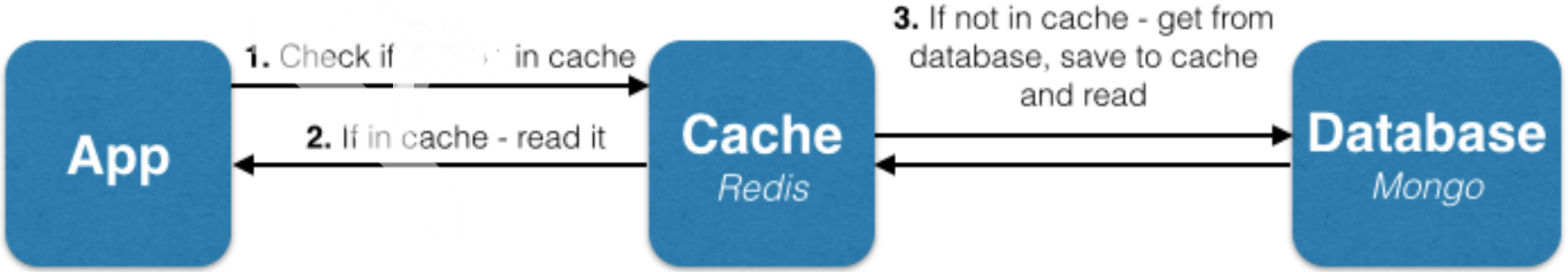
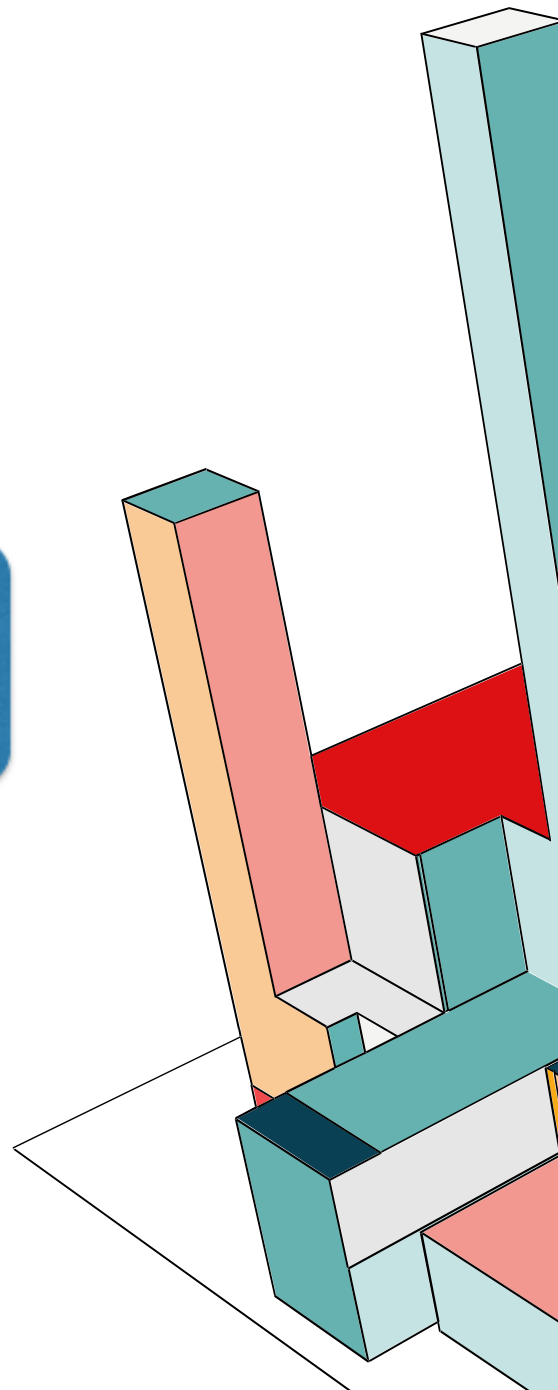
ARCHITECTURE & ETL DATA FLOW

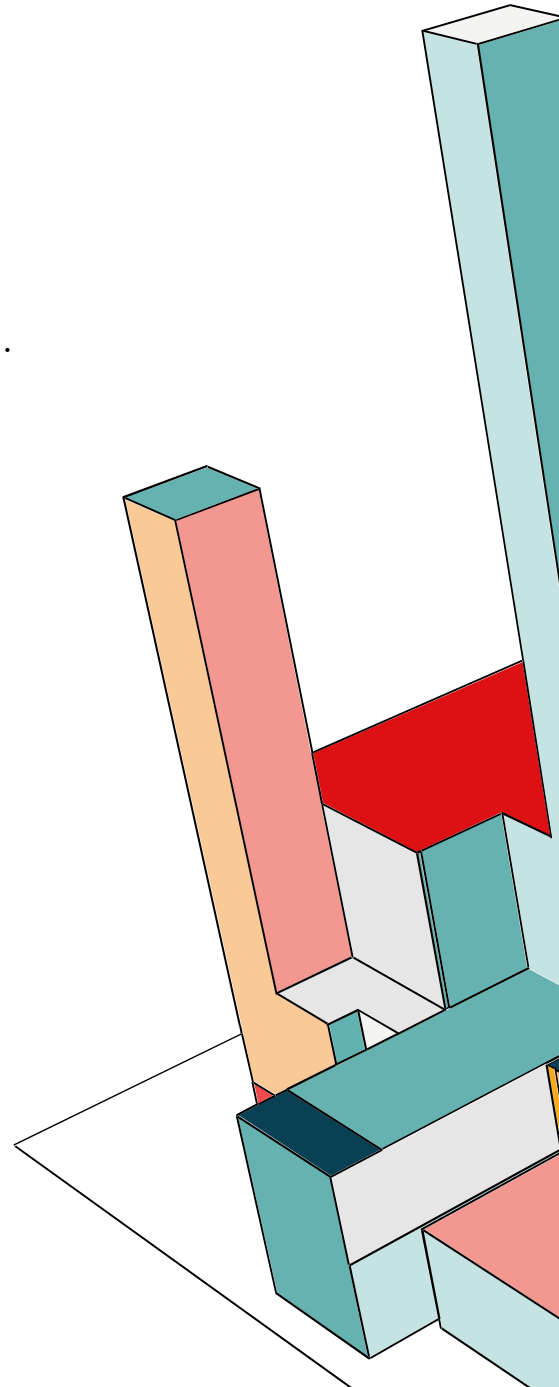




LESSONS LEARNED

- Learned how in-memory caching significantly reduces response time (120 ms → 3 ms).
- Implemented three strategies: Cache-Aside, Read-Through, and TTL Expiration.
- Understood how cache consistency, invalidation, and expiration policies affect data accuracy.
- Gained hands-on practice integrating multiple databases (Redis + MongoDB) in Node.js.

Test Type	Cache Condition	Response Time (ms)	Observation
Cache-Aside	Cache Miss (1st request)	104.248ms	Data fetched from MongoDB
Cache-Aside	Cache Hit (2nd request)	0.549ms	Data served from Redis cache
TTL Cache	Before Expiration	0.522ms	Fast response from Redis
TTL Cache	After Expiration	100.756ms	Cache expired, data reloaded from MongoDB



CHALLENGES & REFLECTIONS

What Worked Well:

- Redis integration was smooth after connection setup.
- Cache hits greatly improved performance and reduced DB load.
- TTL logic automatically kept data fresh.

What Didn’t Work Well:

- Initial environment issues (require vs import, dotenv logging).
- MongoDB URI connection and Redis PATH setup required troubleshooting.
- TTL cache required re-testing to confirm expiration timing.