Spring Boot 3.5.7 React 18 MongoDB 8.0 Security

Kitchensink User Management

Enterprise-Grade Spring Boot Architecture

PRESENTED BY
Your Name

DURATION
45 Minutes

FOCUS
Architecture & Scale

Project Overview

Key Objectives

Migration of legacy Jakarta EE to modern Spring Boot architecture.

- Modernize legacy codebase
- Implement enterprise security (JWT/Encryption)
- Ensure production-ready code quality
- Build scalable, maintainable architecture

Application Scope

Comprehensive user lifecycle management system.

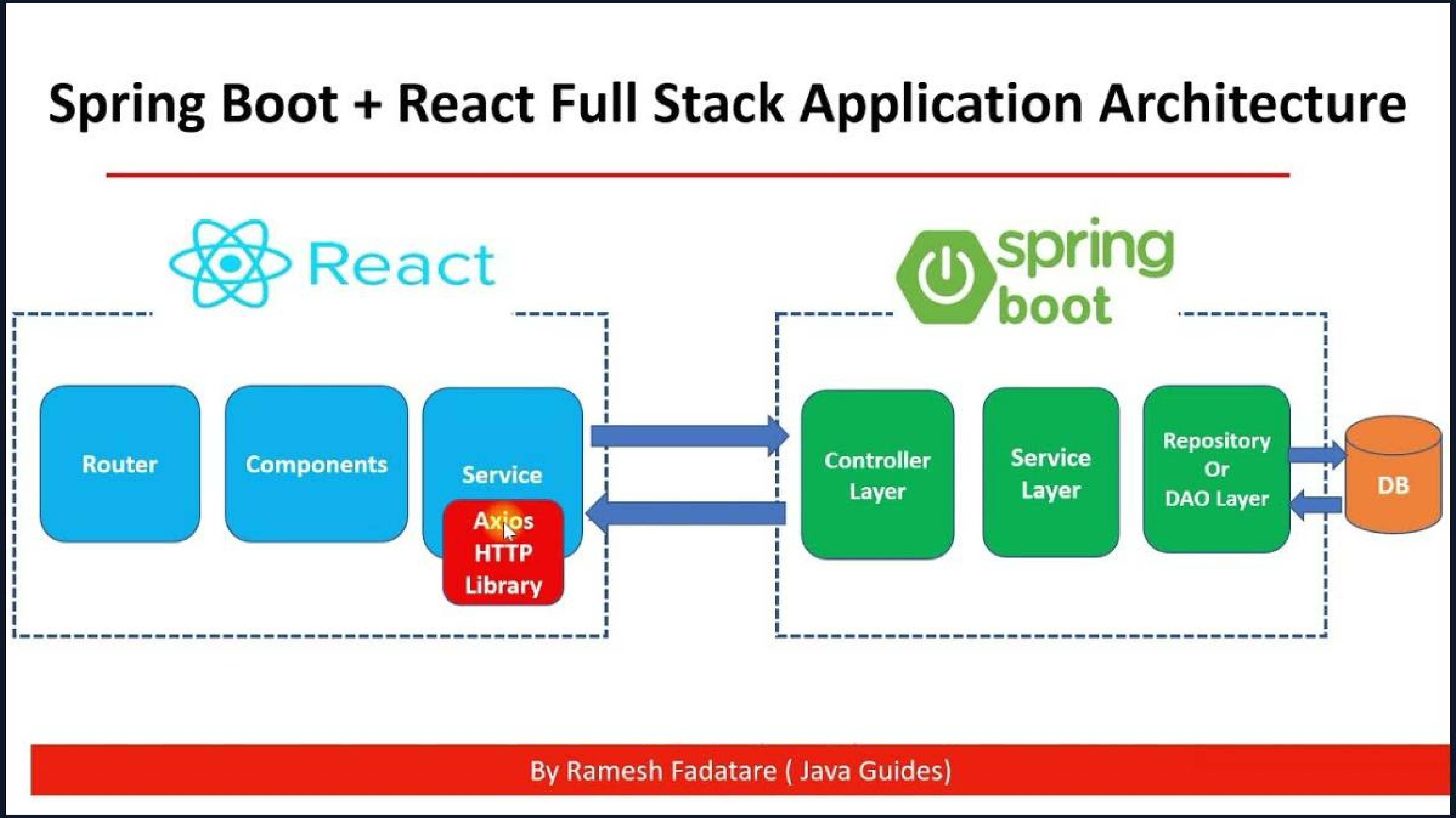
- User Registration & Management
- OTP-based Authentication (Passwordless)
- Role-Based Access Control (RBAC)
- Profile Workflow & Audit Logging

System Architecture

Layered Pattern

- **Frontend:** React 18 (Port 3000)
- **Security:** Filter Chain (CORS, JWT, RateLimit)
- **Controller:** API Routing (Auth, Profile, Admin)
- **Service:** 11 Core Services (Business Logic)
- **Data:** 6 Repositories, MongoDB Collections

Design Principles: Separation of Concerns, Dependency Injection, Event-Driven Architecture.



Technology Stack

Component	Technology	Why Chosen?
Framework	Spring Boot 3.5.7 / Java 21	Latest LTS support, robust ecosystem
Database	MongoDB 8.0+	Flexible schema for user profiles
Security	Spring Security + JWT	Stateless auth for horizontal scaling
Caching	Caffeine (In-Memory)	Sub-millisecond access times
Frontend	React 18 + Axios	Modern UI, component-based architecture
Quality	JUnit 5 + Mockito + JaCoCo	Ensures 100% test coverage

Security Architecture

Request Filter Chain

Sequential processing: CORS → CorrelationId → RateLimit → RequestLog → ApiKey → JWT.

Authentication

OTP-based (SHA-256 hashed, 5-min TTL) and JWT (Stateless, Role Claims) for API access.

Data Protection (PII)

AES encryption at rest for email/phone. Hash-based indexing allows searching without decryption.

Rate Limiting

API Level: 60 req/min. OTP Level: 1000 attempts/15 min. Returns HTTP 429 on breach.

Database Design

MongoDB Collections

users: Stores PII (Encrypted + Hashed), Status, Dates.

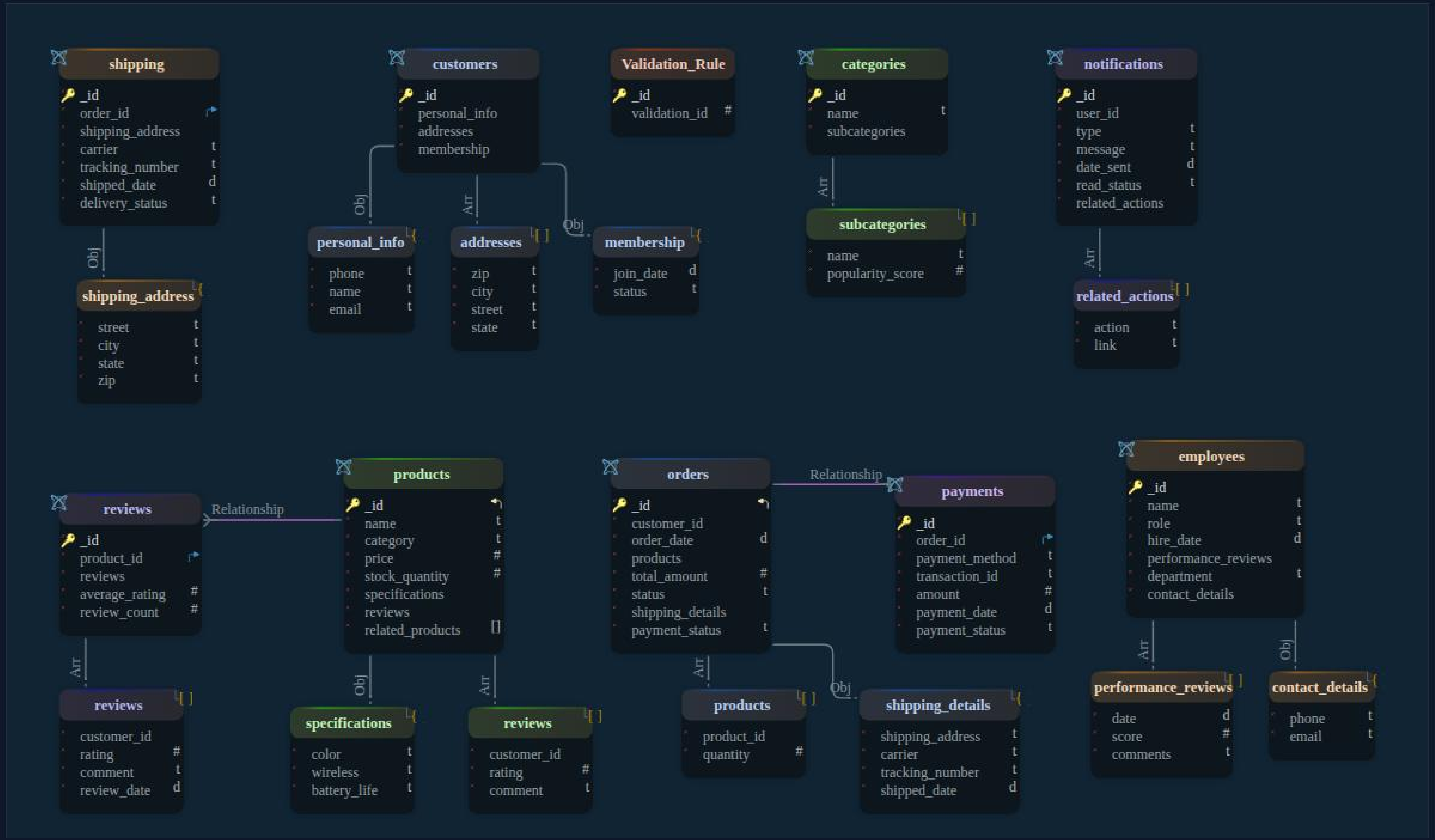
roles / user_roles: RBAC mapping.

otps: Hashed OTPs with TTL indexes.

update_requests: Tracks profile changes for Admin approval.

audit_logs: Entity changes, correlation IDs, timestamps.

Indexing Strategy:
Unique (emailHash), Compound (userId+roleId), TTL (OTP expiration).



RESTful API Design

Core Endpoints

Base URL: `/kitchensink/v1/`

- `POST /auth/login/request-otp`
- `PUT /profile/{userId}` (Triggers approval)
- `GET /admin/users?useCursor=true`
- `POST /admin/update-requests/{id}/approve`

Design Patterns

- **Pagination:** Supports Offset and Cursor-based (for deep scrolling).
- **Error Handling:** Global Exception Handler, Structured JSON responses.
- **Standard Response:**

```
{
  "success": true,
  "data": {...},
  "correlationId": "UUID"
}
```

Key Features

OTP Auth

Passwordless login via email. Hashed storage and strict rate limiting.

Approvals

Strict workflow: Admin approval required for sensitive profile updates.

Audit Log

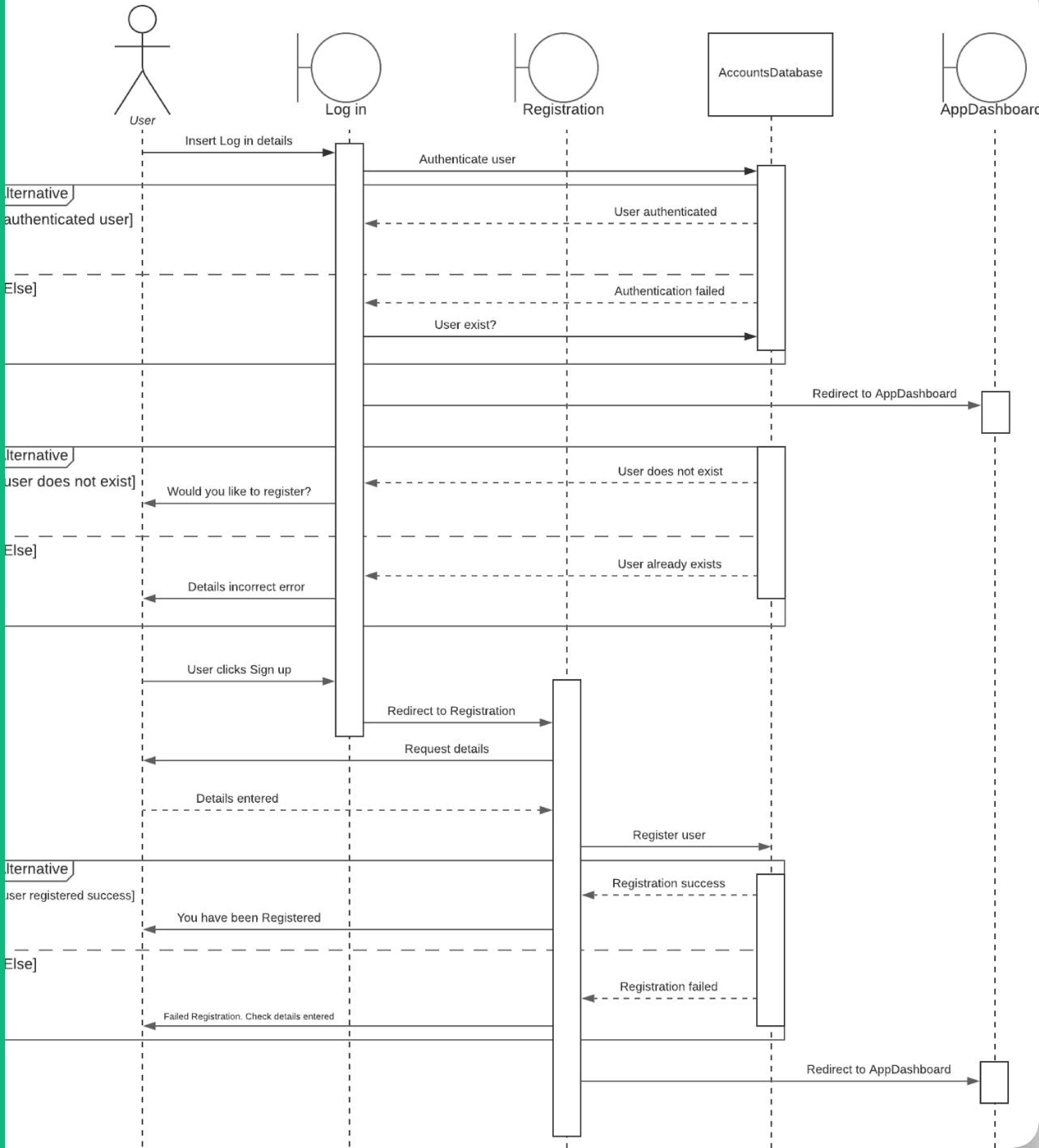
Async tracking of all field-level changes via MongoDB event listeners.

Cursor Paging

Optimized for large datasets. O(1) complexity for deep pagination.

Login Request Flow

1. **Filter Chain:** RateLimit → Auth Checks (Skipped for public endpoint).
2. **Controller:** `AuthController` receives request.
3. **Service:** `OtpService` generates 6-digit code.
4. **Encryption:** SHA-256 Hashing of OTP.
5. **Persistence:** Store hashed OTP in MongoDB.
6. **Async Action:** `EmailService` sends email (Non-blocking).
7. **Response:** Return 200 OK immediately.



Caching Strategy

Configuration

Engine: **Caffeine Cache** (High-performance, In-memory).

- **User Cache:** Key=UserId, TTL=5 min.
- **Role Cache:** Key=RoleId, TTL=10 min.
- Max Size: 1000 entries (LRU Eviction).

Operations

- **Write-Through:** Updates DB and Cache simultaneously.
- **Hit Rate Goal:** >80% to reduce DB load.
- **Invalidation:** Auto-expiration (TTL) + Manual eviction on profile updates.
- **Speed:** Sub-millisecond access for Auth checks.

Testing & Quality Assurance

100%

Code Coverage

Line, Branch, & Method (JaCoCo)

150+

Total Test Cases

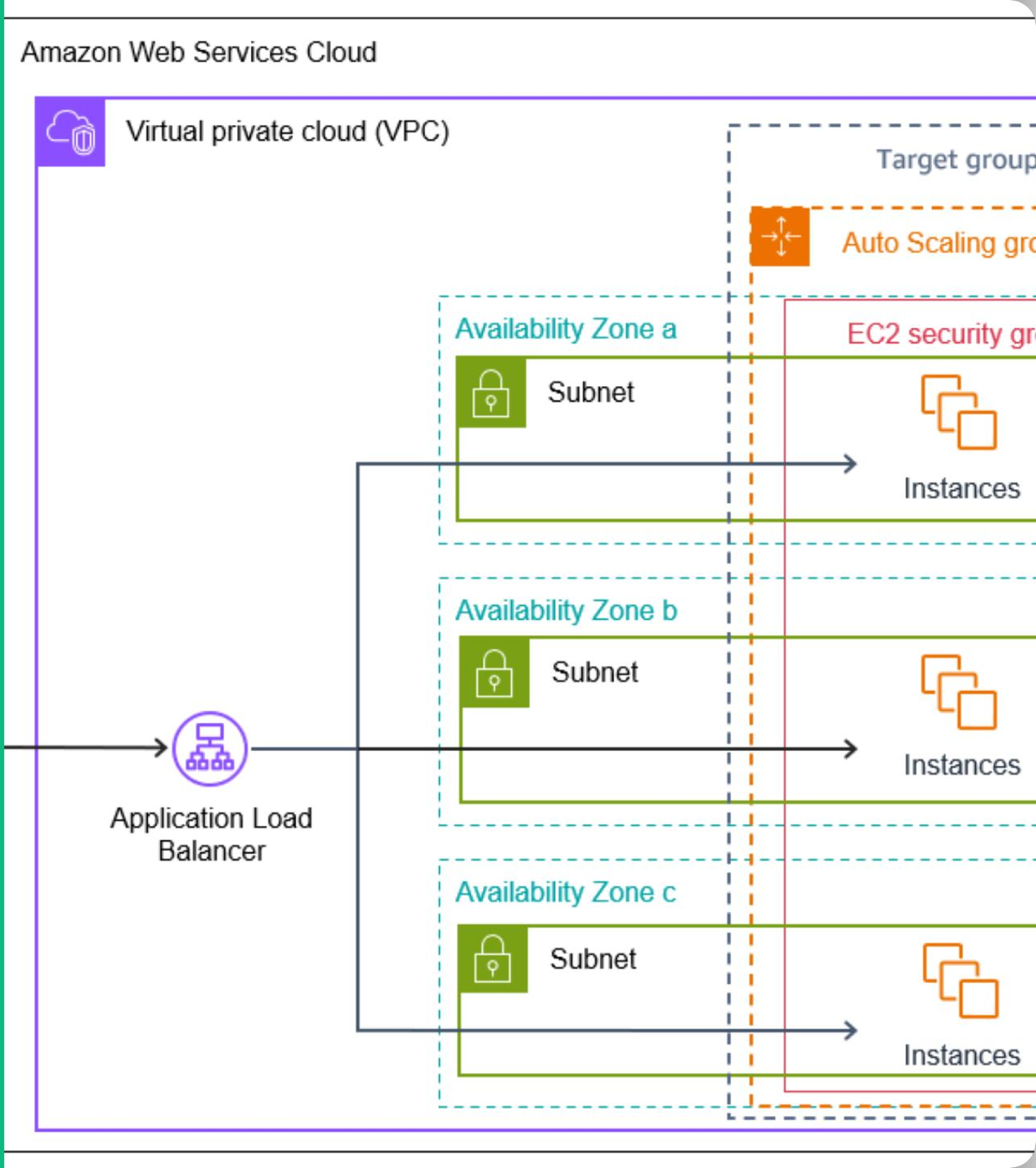
Unit, Integration, & Security Tests

Scaling to 100M+

Phase 1: Horizontal (0-1M)
Load Balancer + Stateless App Instances + DB Replica Set. Replace Caffeine with Redis.

Phase 2: Sharding (1M-10M)
MongoDB Sharding (Key: userId). Compound & Partial Indexes.

Phase 3: Microservices (10M+)
Decompose services (Auth, User, Notif). API Gateway + Kafka for Async.



Performance Optimizations

Cursor-based Pagination

Replaces Offset pagination. Complexity reduces from O(n) to O(1) for deep pages.

Async Processing

Email sending and Audit logging are non-blocking (@Async), reducing API latency.

Smart Indexing

Compound indexes for frequent queries. Hash-based indexes for encrypted PII lookups.

Connection Pooling

Optimized pools for MongoDB and HTTP clients to handle concurrent load.

Key Learnings

PII Strategy

Hash-based indexing solves the dilemma of securing PII while maintaining search capability.

Event Audit

Decoupling audit logs via event listeners ensures zero performance overhead on main flow.

Filter Order

Explicit ordering (RateLimit → Auth) is paramount for robust security.

Stateless Auth

JWT is the foundation for simple, cost-effective horizontal scaling.

Challenges & Solutions

Challenge	Solution	Result
Deep Pagination Latency	Implemented **Cursor-based Pagination**	100x faster for deep pages
Searching Encrypted PII	**Hash-based Indexing** + Encryption	Secure + Queryable
Audit Blocking API	**MongoDB Event Listeners + @Async**	Zero performance impact
Cache Consistency	**Write-through + TTL Strategy**	Balanced speed & freshness

Future Enhancements

SHORT TERM (3-6 MO)

- Multi-factor Authentication (MFA)
- Elasticsearch Integration
- Real-time WebSockets

MEDIUM TERM (6-12 MO)

- Microservices Decomposition
- API Gateway Implementation
- Advanced Analytics Dashboards

LONG TERM (12+ MO)

- AI Fraud Detection
- Blockchain Immutable Logs
- GraphQL API Layer

Q & A

Production-Ready. Scalable. Secure.
Designed to scale from zero to 100 million users.

Thank You!

github.com/your-repo | contact@kitchensink.com

Image Sources

https://i.ytimg.com/vi/LCT4LPm5dnI/maxresdefault.jpg?sqp=-oaymwEmCIAKENAF8quKqQMa8AEB-AHUBoAC4AOKAgwIABABGH8gGSgTMA8=&rs=AOn4CLBUoXhZdvyN8cFxnjZx4b5WcCW-VA

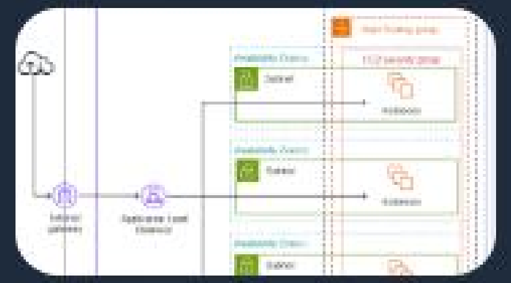
Source: www.youtube.com

https://dbschema.com/blog/mongodb/mongodb-database-diagram/mongodb-diagram.svg

Source: dbschema.com

https://i.sstatic.net/RTchP.png

Source: stackoverflow.com



https://docs.aws.amazon.com/images/autoscaling/ec2/userguide/images/elb-tutorial-architecture-diagram.png

Source: docs.aws.amazon.com