Secure NoSQL Database Design for a Social Media Application

Project Report

Database Security Fundamentals (UGCSA203)

Bachelor of Computer Application

PROJECT GUIDE:

ZAHID AHMED

SUBMITTED BY:

MD SHABBIR (23CSA2BC214)
MOHAMMAD DANISH (23CSA2BC323)
KUNAL KUMAR (23CSA2BC230)
MUKESH KUMAR (23CSA2BC280)

MAR,2025

VIVEKANANDA GLOBAL UNIVERSITY

ACKNOWLEDGEMENT

I have taken this opportunity to express my gratitude and humble regards to the Vivekananda Global University to provide an opportunity to present a project on the "Secure NoSQL Database Design for a Social Media Application" which is a Database Security Fundamentals Project.

*I would also be thankful to my project guide **ZAHID AHMED** to help me in the completion of my project and the documentation.I have taken efforts in this project but the success of this project would not be possible without their support and encouragement.*

I would like to thanks our principal sir "Dr.Surendra Yadav" to help us in providing all the necessary books and other stuffs as and when required .I show my gratitude to the authors whose books has been proved as the guide in the completion of my project I am also thankful to my classmates and friends who have encouraged me in the course of completion of the project.

Thanks

MD SHABBIR (23CSA2BC214)
MOHAMMAD DANISH (23CSA2BC323)
KUNAL KUMAR (23CSA2BC230)
MUKESH KUMAR (23CSA2BC280)

Place: **Jaipur**

Date: **31/03/25**

DECLARATION

We hereby declare that this Project Report titled “Secure NoSQL Database Design for a Social Media Application” submitted by us and approved by our project guide, to the Vivekananda Global University, Jaipur is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project Group

Student Name:	**MD SHABBIR (23CSA2BC214)** **MOHAMMAD DANISH (23CSA2BC323)** **KUNAL KUMAR (23CSA2BC230)** **MUKESH KUMAR (23CSA2BC280)**
Project Guide:	**ZAHID AHMED**

Table of Contents

1. Abstract
2. Introduction
    1. 2.1 Background
    2. 2.2 Objectives
    3. 2.3 NoSQL Overview
3. Database Schema Design
    1. 3.1 Collections and Document Structure
        1. 3.1.1 Users Collection
        2. 3.1.2 Posts Collection
        3. 3.1.3 Comments Collection
        4. 3.1.4 Likes Collection
    2. 3.2 Flexibility of NoSQL Data Models
4. Security Implementation
    1. 4.1 Encryption and Hashing
    2. 4.2 Role-Based Access Control (RBAC)
    3. 4.3 Audit Logging and Threat Monitoring
    4. 4.4 Schema Validation and Data Integrity
    5. 4.5 Privacy and Compliance (GDPR/CCPA)
5. Performance and Trade-offs
    1. 5.1 Denormalization vs. Consistency
    2. 5.2 Encryption Overhead
6. Summary
7. Conclusion
8. References
9. Appendices
    1. A. ER Diagrams
    2. B. Example Queries
    3. C. Screenshots of Implementation

Abstract

This project designs and implements a secure NoSQL database schema for a social media application using MongoDB, emphasizing database security fundamentals. The schema supports core features like user profiles, posts, comments, and likes while addressing critical security challenges such as data breaches, unauthorized access, and regulatory compliance. Key security measures include client-side field-level encryption, role-based access control (RBAC), audit logging, and schema validation. The report evaluates the trade-offs between NoSQL's flexibility and security requirements, demonstrating how modern databases can balance scalability with robust protection mechanisms. This work serves as a practical guide to secure NoSQL design for evolving applications.

Introduction

2.1 Background

Social media platforms generate vast unstructured data, requiring scalable databases like MongoDB. However, NoSQL's schema-less nature introduces security risks such as injection attacks, exposed sensitive data, and compliance gaps. This project addresses these challenges through a security-first database design.

2.2 Objectives

1. Design a NoSQL schema for a social media app with users, posts, comments, and likes.
2. Integrate security features: encryption, RBAC, audit logging, and GDPR compliance.
3. Analyze performance-security trade-offs in NoSQL systems.

2.3 NoSQL Overview

NoSQL databases prioritize horizontal scalability and flexible schemas, avoiding rigid joins and tables. While ideal for social media's dynamic data, this flexibility demands proactive security design to mitigate risks like data leakage or inconsistent authorization.

3. Database Schema Design

3.1 Collections and Document Structure

3.1.1 Users Collection

```
{
  _id: ObjectId("507f191e810c19729de860ea"),
  username: "john_doe", // Unique
  email: "john@example.com", // Encrypted
  password_hash: "$2a$10$...", // Bcrypt hash
  role: "user", // RBAC: "user", "admin", "moderator"
  security_question: {
    question: "Favorite book?",
    answer_hash: "$2a$10$..."
  },
  privacy_settings: {
    is_public: true,
    hide_email: false
  },
  failed_login_attempts: 0,
  last_login: ISODate("2023-10-01T12:00:00Z")
}
```

3.1.2 Posts Collection

```
{
  _id: ObjectId("65bd1e29c7a8f12a7c8e3d1a"),
  user_id: ObjectId("507f191e810c19729de860ea"),
  content: "Hello, world!",
  timestamp: ISODate("2023-10-01T12:05:00Z"),
  likes_count: 25, // Denormalized counter
  comments_count: 5
}
```

3.1.3 Comments Collection

```
{
  _id: ObjectId("65bd1e29c7a8f12a7c8e3d1b"),
  user_id: ObjectId("507f191e810c19729de860ea"),
  post_id: ObjectId("65bd1e29c7a8f12a7c8e3d1a"),
  content: "Great post!",
  timestamp: ISODate("2023-10-01T12:10:00Z"),
  likes_count: 2
}
```

3.1.4 Likes Collection

```
{
  _id: ObjectId("65bd1e29c7a8f12a7c8e3d1c"),
  user_id: ObjectId("507f191e810c19729de860ea"),
  target_type: "post", // "post" or "comment"
  target_id: ObjectId("65bd1e29c7a8f12a7c8e3d1a"),
  timestamp: ISODate("2023-10-01T12:15:00Z")
}
```

3.2 Flexibility of NoSQL Data Models

- Denormalization: Embedding `username` in posts/comments avoids joins.
- Polymorphism: The `likes` collection supports posts and comments via `target_type`.
- Schema Evolution: Add fields like `hashtags` or `media_type` without downtime.

4. Security Implementation

4.1 Encryption and Hashing

- Client-Side Field-Level Encryption: Encrypt `email` and `security_question.answer` using MongoDB's encryption libraries.
- Bcrypt Hashing: Securely hash passwords with a cost factor of 12.

4.2 Role-Based Access Control (RBAC)

- Define roles in a `permissions` collection:

```
{
  role: "admin",
  can_delete_posts: true,
  can_ban_users: true
}
```

- Use middleware to enforce permissions (e.g., only admins can delete posts).

4.3 Audit Logging

```
// Audit Logs Collection
{
  _id: ObjectId("65bd1e29c7a8f12a7c8e3d1d"),
  user_id: ObjectId("507f191e810c19729de860ea"),
  action: "login",
  ip_address: "192.168.1.1",
  timestamp: ISODate("2023-10-01T12:20:00Z")
}
```

4.4 Schema Validation

Enforce data integrity for the `users` collection:

```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      required: ["username", "email", "password_hash"],
      properties: {
        email: { bsonType: "string", pattern: "^[a-zA-Z0-9._%+-]+@[a-
zA-Z0-9.-]+\\.[a-zA-Z]{2,}$" },
        role: { enum: ["user", "admin", "moderator"] }
      }
    }
  }
});
```

4.5 Privacy and Compliance

- GDPR Compliance: Allow users to anonymize data via `pseudonym` fields.
- Privacy Settings: Let users toggle visibility of profiles/emails.

5. Performance and Trade-offs

5.1 Denormalization vs. Consistency

- Pros: Faster reads for `likes_count` and `comments_count`.
- Cons: Requires transactions to sync counters with the `likes`/`comments` collections.

5.2 Encryption Overhead

- Client-Side Encryption adds ~10-15% latency but ensures data confidentiality.
- Mitigate with hardware acceleration or optimized key management.

6. Summary

The project designed a secure, scalable NoSQL schema for a social media app, prioritizing:

- Security: Encryption, RBAC, audit logs, and GDPR compliance.
- Performance: Denormalized counters and indexed queries.
- Flexibility: Polymorphic relationships and schema evolution.

7. Conclusion

This project demonstrates that NoSQL databases like MongoDB can meet rigorous security standards while retaining their core advantages of scalability and flexibility. By integrating encryption, RBAC, and audit logging, the design mitigates risks like unauthorized access and data breaches. Future enhancements could include rate-limiting, anomaly detection in audit logs, and multi-cloud backups. This work underscores the importance of embedding security into database design from the outset, aligning with both academic principles and industry best practices.

8. References

MongoDB Security Best Practices.

GDPR Compliance Guidelines.

Bcrypt Hashing Algorithm Documentation.

9. Appendices

A. ER Diagrams

```
+-------------+          +-------------+
|   Users     |          |   Posts     |
+-------------+          +-------------+
| _id (PK)    |◀---1:N---| user_id (FK)|
| username    |          |   content   |
| email 🔒     |          +-------------+
+-------------+                  |
     |                           |
     | 1:N                       | 1:N
     ▼                           ▼
+-------------+           +-------------+
|  Comments   |           |   Likes     |
+-------------+           +-------------+
| post_id (FK)|◀-----N:1-| target_id   |
+-------------+           | target_type |
                          +-------------+
                              |
                              | N:1
               +----------------------------+
               ▼                            ▼
         +-------------+          +-------------+
         |   Posts     |          |  Comments   |
         +-------------+          +-------------+
```

B. Example Queries

- Detect Brute-Force Attacks:

```
db.users.find({
  failed_login_attempts: { $gt: 5 },
  last_login: { $gte: new Date(Date.now() - 3600 * 1000) }
});
```