Hospital Management System Lab Report

---

1. Redis Integration

Objective:

- Integrate Redis into the Hospital Management System and perform basic operations such as setting and retrieving values, working with hash structures, and managing lists.

Overview:

- Redis was successfully integrated into the Hospital Management System. The setup involved configuring the Redis connection within the Spring Boot application.
- Basic operations were tested to ensure the connection and functionality were working as expected:
    - **Set and Get Operations:** These were used to store and retrieve data, such as storing the hospital's name and retrieving it to confirm data integrity.
    - **Hash Operations:** Redis hashes were utilized to store and manage patient records in a structured format.
    - **List Operations:** Redis lists were employed to manage queues, such as patient waiting lists, providing efficient access and manipulation of list data.

---

2. Spring Data Redis

Objective:

- Use Spring Data Redis to enable repository-based access to Redis within the Hospital Management System.

Overview:

- Spring Data Redis was employed to streamline data access and management through repositories, abstracting the lower-level Redis operations.
- Repository integration facilitated the management of hospital-related data, including patient records and appointment scheduling, directly through Redis with improved efficiency and ease of use.
- The system demonstrated effective read and write operations, leveraging the full potential of Redis as a high-performance, in-memory data store.

---

3. MongoDB Integration

Objective:

- Transition from relational databases to NoSQL by setting up a MongoDB connection and creating collections within the Hospital Management System.

Overview:

- MongoDB was integrated into the system as an alternative to traditional relational databases.
- The integration involved configuring the MongoDB connection and creating collections tailored to the hospital's needs, such as collections for patient records, medical history, and doctor profiles.
- The transition provided the system with greater flexibility in managing unstructured and semi-structured data, aligning with the evolving needs of modern healthcare management.

4. Spring Data MongoDB

Objective:

- Use Spring Data MongoDB to perform CRUD (Create, Read, Update, Delete) operations on documents within the Hospital Management System.

Overview:

- Spring Data MongoDB was utilized to manage hospital data in a NoSQL environment, enabling efficient CRUD operations on documents.
- Patient information, appointment details, and other essential data were stored and managed within MongoDB collections, ensuring the system could scale and adapt to complex data structures.
- The system successfully demonstrated the ability to perform robust data management tasks, such as updating patient records and deleting outdated information, all within the NoSQL database.

---

5. Querying NoSQL Databases

Objective:

- Perform basic and advanced queries on Redis and MongoDB within the Hospital Management System.

Overview:

- Queries were performed on both Redis and MongoDB to extract relevant hospital data.
    - **Redis Queries:** Basic queries, such as retrieving specific patient details or list-based queries, were executed to manage real-time data effectively.
    - **MongoDB Queries:** Both basic and advanced queries were employed to filter patient records, aggregate data for reporting purposes, and retrieve specific medical history information.
- The system demonstrated proficiency in querying NoSQL databases, ensuring that data retrieval is efficient and meets the operational needs of the hospital.

Conclusion:

- The integration of Redis and MongoDB into the Hospital Management System significantly enhanced the system's performance and scalability.
- Spring Data Redis and Spring Data MongoDB provided seamless data management capabilities, allowing the system to handle complex data structures and high-volume operations effectively.
- The successful querying of NoSQL databases ensured that the system could meet the dynamic data retrieval needs of a modern hospital, supporting both basic and advanced use cases.

This report illustrates the smooth transition from traditional relational databases to NoSQL solutions, highlighting the improvements in performance, flexibility, and data management within the Hospital Management System.