**Case Study Document: Spring Boot and Couchbase**

**Table of Contents**

1. Introduction

- 1.1 Purpose

- 1.2 Scope

- 1.3 Technologies Used

2. Business Scenario

- 2.1 Background

- 2.2 Problem Statement

- 2.3 Objectives

3. System Architecture

- 3.1 High-Level Overview

- 3.2 Components

- 3.3 Data Flow

4. Features

- 4.1 User Management

- 4.2 Data Storage and Retrieval

- 4.3 CRUD Operations

- 4.4 Indexing and Querying

5. Implementation Steps

- 5.1 Setting up Couchbase Server

- 5.2 Creating a Spring Boot Project

- 5.3 Configuring Spring Boot and Couchbase Integration

- 5.4 Implementing User Management

- 5.5 Handling Data Storage and Retrieval

- 5.6 Implementing CRUD Operations

- 5.7 Setting Up Indexes and Querying

**1. Introduction**

**1.1 Purpose**

The purpose of this case study is to demonstrate the integration of Spring Boot, a powerful Java-based framework for building web applications, and Couchbase, a NoSQL database. The assignment aims to showcase the implementation of essential features using this technology stack.

**1.2 Technologies Used**

- Java 8+

- Spring Boot 2.x

- Couchbase Server 6.x

- Maven or Gradle for dependency management

- IDE of choice (e.g., IntelliJ, Eclipse)

**2. Business Scenario**

**2.1 Background**

The assignment is based on a scenario where a company requires a web application to manage user data. The data consists of users' basic information, such as name, email, and age. The application should support storing, retrieving, updating, and deleting user records.

**2.2 Problem Statement**

The medical consultation company needs an efficient and scalable solution to manage doctor’s data. Traditional relational databases may not be the best fit for the requirements, as the data schema may change frequently, and the volume of data may grow rapidly. Therefore, the company has decided to explore NoSQL databases, specifically Couchbase, to address these challenges.

**2.3 Objectives**

The main objectives of this case study are:

1. Implement a Spring Boot application integrated with Couchbase as the backend database.

2. Create RESTful API endpoints to perform CRUD operations on user data.

3. Demonstrate the use of Couchbase's flexible schema and document storage capabilities.

4. Validate and test the application for functionality and performance.

**3. System Architecture**

**3.1 High-Level Overview**

The system will follow a typical three-tier architecture:

1. Presentation Layer: Handles incoming HTTP requests and returns responses to clients.

2. Business Logic Layer: Contains application logic and coordinates data access.

3. Data Access Layer: Manages interaction with the Couchbase NoSQL database.

**3.2 Components**

The major components of the system include:

- Controller Layer: Exposes RESTful API endpoints for user data management.

- Service Layer: Implements business logic and interacts with the Data Access Layer.

- Data Access Layer: Communicates with the Couchbase NoSQL database.

**3.3 Data Flow**

The data flow within the system is as follows:

1. Clients (web or mobile applications) send HTTP requests to the API endpoints.

2. The Controller Layer receives the requests and validates the inputs.

3. The Service Layer processes the requests, performs business logic, and communicates with the Data Access Layer for data retrieval or storage.

4. The Data Access Layer interacts with the Couchbase database to store or retrieve data.

5. Responses are sent back through the Controller Layer to the clients.

**4. Features**

The web application will offer the following features:

**4.1 User Management**

- Create new doctor’s records with name, email, and age.

- Retrieve doctor’s records by ID or email.

- Update existing Doctor information.

- Delete Doctor records.

**4.2 Data Storage and Retrieval**

- Demonstrate the use of Couchbase's flexible schema for storing user data as JSON documents.

- Retrieve doctor data from Couchbase using key-value access.

**4.3 CRUD Operations**

- Implement Create, Read, Update, and Delete operations using Spring Boot and Couchbase.

- Validate inputs and handle exceptions gracefully.

4.4 Indexing and Querying

- Create secondary indexes on user data fields for efficient querying.

- Perform simple N1QL queries to retrieve user records based on specific criteria.

**5. Implementation Steps – To be filled and submitted by the learner**

**5.1 Setting up Couchbase Server**

<<Explain the steps to install and configure Couchbase Server for the assignment.>>

**5.2 Creating a Spring Boot Project**

<<Steps for creating a new Spring Boot project using the Spring Initializr or any other method.>>

**5.3 Configuring Spring Boot and Couchbase Integration**

<<Instructions on adding the required dependencies and configuration to integrate Spring Boot with Couchbase.>>

**5.4 Implementing User Management**

<<Step-by-step guide to creating the necessary API endpoints and services for user management.>>

**5.5 Handling Data Storage and Retrieval**

<<Demonstrate how to store user data as JSON documents in Couchbase and retrieve it using key-value access.>>

**5.6 Implementing CRUD Operations**

<<Walk through the implementation of Create, Read, Update, and Delete operations using Spring Boot and Couchbase.>>

**5.7 Setting Up Indexes and Querying**

<<Explain the process of creating secondary indexes on user data fields and performing simple N1QL queries>>