Senior Software Developer Task

# Task Overview

Welcome to the technical task for the Senior Software Developer. This task is designed to evaluate your skills in SQL, Java, Spring Boot, and AngularJS/ReactJS, as well as your ability to handle large datasets and refactor existing code. Please read each section carefully and complete the tasks to the best of your ability.

# Part 1: SQL Schema Design

## Task 1: Create Entry Tables and Java Classes

Based on the provided SQL script and the requirement of having 5 types of entries linked to `entryId` in the `TEntryTransaction` table (See the Reference):  
  
1. Design 5 possible tables to represent the different types of entries.  
2. Create the related Java classes for these entries.  
3. Explain your design solution, focusing on:  
 - The relationship between `TEntryTransaction` and the entry tables.  
 - How you ensure data integrity and referential integrity.  
 - Any considerations for future scalability or maintainability.

# Part 2: Project Development

## Task 2: Create Project with Spring Boot and AngularJS/ReactJS

Create a project that includes a Spring Boot backend and a frontend using AngularJS/ReactJS/VueJS (or pure JS). The project should support CRUD operations for `TEntryTransaction`.  
  
**I. Spring Boot Backend**  
  
1. Set up a Spring Boot project with dependencies for JPA, MySQL, and Web.  
2. Create entities for `TEntryTransaction` and the five entry types.  
3. Implement repositories for database interactions.  
4. Create REST controllers to handle CRUD operations.  
  
**II. AngularJS/ReactJS Frontend**  
  
1. Set up a project using either AngularJS or ReactJS.  
2. Create components/pages for listing, creating, updating, and deleting `TEntryTransaction` records.  
3. Implement API calls to the Spring Boot backend for each operation.  
4. Ensure proper error handling and form validation.

# Part 3: Refactoring for Large Data

## Task 3: Create Mockup Data and Refactor Solution

Assume we already have a huge `TEntryTransaction` dataset (around 1 billion records). Now, we want to refactor the project to store `TAccountTags` without the `year` attribute.  
  
1. Create Mockup Data: Generate a sample dataset to simulate the large `TEntryTransaction` table.  
2. Refactor Solution: Provide a detailed solution on how to refactor the project to remove the `year` attribute from `TAccountTags`.  
  
**## Solution Components**  
  
1. Database Schema Changes: Describe the changes you would make to the database schema to remove the `year` attribute from `TAccountTags`.  
2. Data Migration: Outline a strategy for migrating existing data to the new schema without causing significant downtime.  
3. Code Refactoring: Detail the necessary changes to the Java codebase to accommodate the updated schema.  
4. Performance Considerations: Explain how you would ensure the refactored system performs efficiently with the large dataset.

# Submission Guidelines

* Please submit your SQL scripts, Spring Boot project, and frontend project as a zipped file or a link to a GitHub repository.
* Include a README file with instructions on how to set up and run the projects.
* For the open question, provide a detailed written solution either as a markdown file or a PDF document.

Good luck, and we look forward to reviewing your work!

# Reference

# Provided DB Script

CREATE TABLE `T\_EntryTransaction` (  
 `id` bigint NOT NULL AUTO\_INCREMENT,  
 `taccId` bigint DEFAULT NULL,  
 `entryId` bigint NOT NULL,  
 `type` varchar(30) NOT NULL,  
 `amount` decimal(20,2) NOT NULL,  
 `transactionDate` date NOT NULL,  
 `fundId` varchar(32) NOT NULL,  
 `date\_created` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 `last\_updated` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb3;  
  
CREATE TABLE `TAccount` (  
 `id` bigint NOT NULL AUTO\_INCREMENT,  
 `accountID` bigint NOT NULL,  
 `pid` bigint DEFAULT NULL,  
 `code` varchar(20) DEFAULT NULL,  
 `name` varchar(255) DEFAULT NULL,  
 `accountClass` varchar(1) DEFAULT NULL,  
 `accountType` varchar(20) DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 CONSTRAINT `FK\_taccount\_pid` FOREIGN KEY (`pid`) REFERENCES `TAccount` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT  
) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb3 STATS\_SAMPLE\_PAGES=256;  
  
CREATE TABLE `TAccountTags` (  
 `id` bigint NOT NULL AUTO\_INCREMENT,  
 `name` varchar(255) DEFAULT NULL,  
 `color` varchar(20) DEFAULT NULL,  
 `type` varchar(50) DEFAULT NULL,  
 `year` smallint DEFAULT NULL,  
 `isSystemLabel` bit(1) DEFAULT NULL,  
 `code` varchar(100) DEFAULT NULL,  
 `item\_order` int DEFAULT NULL,  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb3;  
  
CREATE TABLE `TAccountTag` (  
 `tagID` bigint NOT NULL,  
 `taccountID` bigint NOT NULL,  
 PRIMARY KEY (`taccountID`,`tagID`),  
 CONSTRAINT `FK\_taccountTag\_taccount` FOREIGN KEY (`taccountID`) REFERENCES `TAccount` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,  
 CONSTRAINT `FK\_taccountTag\_taccountTags` FOREIGN KEY (`tagID`) REFERENCES `TAccountTags` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 STATS\_SAMPLE\_PAGES=128;

# Provided Java Class

public class TEntry {  
 private Long id;  
 private BigDecimal amount;  
 private BigDecimal gstAmount;  
}

publicenumTEntryType {  
 BasicBankEntry,  
 DistributionInterest,  
 Dividend,  
 Contribution,  
 Investment  
}