Below is an updated version of the system design specifications incorporating the comparison of the reading composition score to a benchmark score from Aspen reference tables and the addition of an alert to the student’s Aspen account if the score falls below the benchmark. I’ve formatted it in a way that’s ready to be copied into a Microsoft Word document with standard headings and structure for clarity. Let me know if you need further tweaks!

**System Design Specifications: Aspen SIS Assessment Score Calculators**

**1. Overview**

This document outlines the system design for three Java-based processes integrated as stored procedures within the Aspen Student Information System (SIS). These processes calculate reading assessment scores for students and compare them against benchmarks to trigger alerts. The processes are:

1. **Acadience Reading Score Calculator (Process 1)**
2. **Acadience FR Reading Score Calculator (Process 2)**
3. **DIBELS Score Calculator (Process 3)**

Each process follows a similar workflow but applies distinct calculations tailored to the assessment type. The system reads data from the Aspen SIS database, performs calculations, updates scores, compares them to benchmarks, and adds alerts to student accounts as needed.

**2. Objectives**

* Automate the calculation of reading assessment scores based on raw data from Aspen SIS.
* Ensure scores are calculated only for assessments completed within the last 5 days.
* Update existing fields with new values if differences are detected.
* Compare reading composition scores to benchmarks and flag students with below-benchmark scores by adding alerts to their Aspen accounts.

**3. System Components**

**3.1. Java Classes**

* **Class 1**: ProcessAcadienceAssessment
  + Purpose: Calculates Acadience reading scores using calculation method 1 and compares to benchmarks.
* **Class 2**: ProcessIdapelAssessment
  + Purpose: Calculates Acadience reading scores using calculation method 2 and compares to benchmarks.
* **Class 3**: DibelsScoreCalculator
  + Purpose: Calculates DIBELS-specific reading scores and compares to benchmarks.

**3.2. Database**

* **Source Table**: Aspen Student Assessments Table
  + Contains raw assessment data, including:
    - Assessment ID
    - Student ID (foreign key to student table)
    - Completion timestamp
    - Raw values for accuracy, maze, and reading composition
    - Calculated score fields (e.g., accuracy percentage, maze percentage, reading composition score)
* **Reference Table**: Aspen Reference Tables
  + Contains benchmark scores for reading composition, indexed by assessment type or grade level.
* **Student Table**: Aspen Student Table
  + Stores student records and includes an alert field (e.g., text or flag) to indicate below-benchmark performance.

**3.3. Integration**

* **Aspen SIS Stored Procedures**: Each Java class is registered as an executable stored procedure.
* **Execution Trigger**: Procedures are assumed to run periodically (e.g., daily via Aspen scheduler).

**4. Process Workflow (Common to All Three Classes)**

**4.1. Input**

* **Assessment IDs**: List of assessment records from the Aspen Student Assessments Table.
* **Filter Criteria**: Assessments completed within the past 5 days (e.g., based on March 07, 2025).
* **Benchmark Scores**: Retrieved from Aspen Reference Tables based on assessment type.

**4.2. Steps**

1. **Query Database**:
   * Retrieve all assessment records where completion\_timestamp >= (current\_date - 5 days).
   * Select relevant fields: raw values, student ID, assessment type.
2. **Process Each Assessment**:
   * For each assessment record:
     + Read raw values.
     + Calculate:
       - Accuracy percentage
       - Maze percentage
       - Reading composition score (specific to each process, see Section 5).
3. **Compare and Update Scores**:
   * Check if calculated fields exist in the record.
   * If fields exist and new values differ, overwrite with new values.
   * If fields are empty, insert new values.
4. **Benchmark Comparison**:
   * Retrieve the benchmark score for the reading composition from the Aspen Reference Tables (based on assessment type or grade).
   * Compare the calculated reading composition score to the benchmark.
   * If the score is below the benchmark:
     + Add an alert to the student’s record in the Aspen Student Table (e.g., “Below Benchmark: Reading Composition”).
5. **Commit Changes**:
   * Update the Aspen Student Assessments Table with calculated scores.
   * Update the Aspen Student Table with alerts where applicable.

**4.3. Output**

* Updated records in the Aspen Student Assessments Table with calculated scores.
* Alerts added to the Aspen Student Table for students scoring below benchmarks.

**5. Calculation Logic (Differentiated by Process)**

**5.1. Acadience Reading Score Calculator (Process 1)**

* **Accuracy Percentage**: [Insert formula, e.g., (correct\_responses / total\_responses) \* 100]
* **Maze Percentage**: [Insert formula, e.g., (correct\_maze\_answers / total\_maze\_items) \* 100]
* **Reading Composition Score**: [Insert formula, e.g., weighted average]
* **Benchmark Comparison**: Compare reading composition score to reference table value.

**5.2. Acadience Reading Score Calculator (Process 2)**

* **Accuracy Percentage**: [Insert differing formula]
* **Maze Percentage**: [Insert differing formula]
* **Reading Composition Score**: [Insert differing formula]
* **Benchmark Comparison**: Compare reading composition score to reference table value.

**5.3. DIBELS Score Calculator (Process 3)**

* **Accuracy Percentage**: [Insert DIBELS-specific formula]
* **Maze Percentage**: [Insert formula, if applicable]
* **Reading Composition Score**: [Insert DIBELS-specific formula]
* **Benchmark Comparison**: Compare reading composition score to reference table value.

*Note*: Exact formulas are placeholders. Provide specifics if desired.

**6. Assumptions and Constraints**

* **Time Window**: Assessments older than 5 days are ignored.
* **Benchmark Availability**: Reference tables contain up-to-date benchmarks for all assessment types.
* **Alert Mechanism**: The student table supports an alert field or mechanism (e.g., text field or flag).
* **Data Integrity**: Raw values and student IDs are valid and complete.

**7. Technical Details**

**7.1. Pseudo-Code (Generic for All Processes)**

java

public class AssessmentScoreCalculator {

public void calculateScores() {

Date fiveDaysAgo = new Date(System.currentTimeMillis() - 5 \* 24 \* 60 \* 60 \* 1000);

List<Assessment> assessments = queryAssessments(fiveDaysAgo);

for (Assessment assessment : assessments) {

double accuracyPercent = calculateAccuracy(assessment.getRawAccuracyValues());

double mazePercent = calculateMaze(assessment.getRawMazeValues());

double readingCompScore = calculateReadingComp(assessment.getRawValues());

if (differsFromExisting(assessment, accuracyPercent, mazePercent, readingCompScore)) {

updateAssessment(assessment, accuracyPercent, mazePercent, readingCompScore);

}

double benchmarkScore = getBenchmarkScore(assessment.getAssessmentType());

if (readingCompScore < benchmarkScore) {

addStudentAlert(assessment.getStudentId(), "Below Benchmark: Reading Composition");

}

}

commitChanges();

}

private List<Assessment> queryAssessments(Date cutoff) {

// SQL: SELECT \* FROM student\_assessments WHERE completion\_timestamp >= cutoff

}

private double getBenchmarkScore(String assessmentType) {

// SQL: SELECT benchmark\_score FROM reference\_tables WHERE type = assessmentType

}

private void addStudentAlert(String studentId, String alertMessage) {

// SQL: UPDATE student\_table SET alert = alertMessage WHERE student\_id = studentId

}

}

**7.2. Dependencies**

* **Java Version**: [Specify, e.g., Java 8+]
* **Database Driver**: JDBC driver for Aspen SIS database.
* **Aspen SIS API**: Supports stored procedure execution and table updates.

**8. Future Considerations**

* Add logging for score calculations and alert triggers.
* Handle cases where benchmark scores are missing from reference tables.
* Optimize database queries for large datasets (e.g., indexing on completion\_timestamp and student\_id).

**Notes for MS Word Formatting**

* Use **Heading 1** for section titles (e.g., "1. Overview").
* Use **Heading 2** for subsections (e.g., "3.1. Java Classes").
* Use **Code** style or a monospace font (e.g., Consolas) for the pseudo-code.
* Add a title page with project name, date (March 07, 2025), and your name if needed.

**Questions for You**

1. Do you have specific formulas for the calculations or benchmark retrieval logic?
2. How is the alert stored in the student table (e.g., text field, boolean flag)?
3. Any additional formatting preferences for Word?

Let me know how to proceed!