**Objective:**

Design and implement a Spring Boot-based RESTful service that reads a large dataset (in CSV format) containing student details, stores the data in memory, creates an index based on student names to optimize search operations, and exposes APIs to create the index and search for students by name.

**Dataset:**

* Format: CSV file
* Record Count: Approximately **100,000 student records**
* Columns (5 fields):
  + ID (String)
  + Name (String)
  + Age (Integer)
  + Class (String)
  + Grade (String)

**Functional Requirements:**

**✅ 1. Create Index API**

* **Method**: POST
* **Endpoint**: /index
* **Description**:
  + Parses the CSV file.
  + Stores student records **in memory**.
  + Creates an **in-memory index** on the Name field using a suitable data structure (Map<String, List<Student>>) to allow faster search operations.
  + Parsing and indexing should be done using **multithreading** to improve performance.

**✅ 2. Search API**

* **Method**: GET
* **Endpoint**: /search
* **Query Parameter**: name (String), mode (optional: linear or index, default is index)
* **Description**:
  + Searches for student records by name.
  + Supports two search modes:
    - linear: Performs a linear search through all records.
    - index: Uses the in-memory index for optimized search.
  + Returns a list of matching student records.

**Technical Requirements:**

* Use **Java** and **Spring Boot** framework.
* No use of external databases — all data should reside **in memory**.
* Utilize **multithreading** during CSV parsing to improve performance (e.g., via ExecutorService).
* Maintain proper separation of concerns (Controller, Service, Model layers).
* Optional: Log or compare performance between linear search and index-based search.

**Deliverables:**

* Spring Boot project (source code).
* Sample CSV file (or support for uploading/reading file path).
* API documentation (endpoints and usage).
* Optional: ReadMe with steps to run and test the application.

**Evaluation Criteria:**

* Correctness and completeness of the implementation.
* Code structure and clarity.
* Use of appropriate data structures for indexing.
* Effective use of multithreading.
* REST API design following best practices.