Design

For the Students Reporting System, we will use a web-based application that allows the user to add students and their marks for multiple semesters and subjects, and view reports based on the data. We will use open-source design elements to create a user-friendly interface.

The application will have the following screens:

1. Home Screen - This screen will display the options to Add Students, Add Marks, and View Reports.
2. Add Student Screen - This screen will allow the user to enter the details of a new student such as name, roll number, and email.
3. Add Marks Screen - This screen will allow the user to select a student, choose a semester and subject, and enter the marks for that subject.

Reports Screen - This screen will allow the user to view three types of reports:
a. Average Percentage of whole class in recent semester
b. Average marks of Students in a subject

4. c. Top 2 Consistent Students across all semesters (Maximum average marks)

Technologies Used:

1. Elastic - To store the data of students and their marks for easy retrieval and analysis.
2. Kibana - To visualize the data stored in Elastic in the form of reports.
3. Spring Boot - To create the backend of the web application.
4. JSP - To create the frontend of the web application.

Local Setup:

To set up the application locally, follow these steps:

1. Install Elastic and Kibana on your local machine. Follow the installation instructions provided on their respective websites.
2. Create an index in Elastic to store the data. Use the following mapping to define the fields:

```
{
  "mappings": {
    "properties": {
      "name": {
        "type": "text"
      },
      "roll_number": {
        "type": "keyword"
      },
      "email": {
        "type": "text"
      },
      "semester": {
        "type": "integer"
      },
      "english_marks": {
        "type": "integer"
      },
      "maths_marks": {
        "type": "integer"
      },
      "science_marks": {
        "type": "integer"
      }
    }
  }
}
```

3. Create a Spring Boot project using your favorite IDE.
4. Add the necessary dependencies for Elastic and JSP.
5. Create controllers for the Add Student, Add Marks, and View Reports screens.
6. Use JSP to create the frontend of the application.
7. Use the Elastic Java API to connect to the Elastic instance and perform CRUD operations on the index.
8. Use Kibana to create visualizations and dashboards for the reports.

With these steps, you will have a local instance of the Students Reporting System up and running.

I can give you an outline of how to approach the problem and some guidance on the steps involved in implementing such a system.

Data Model

1. The first step is to define a data model for the system. This includes the following entities:

- Student: contains student information such as name, ID, and contact details.

- Semester: represents a specific semester, and contains information such as the semester number, start and end dates.
- Subject: represents a specific subject, and contains information such as the subject name and code.
- Marks: represents the marks obtained by a student in a specific subject for a particular semester.

Implement Data Storage

2. The next step is to implement the data storage mechanism. You can use any database of your choice such as MySQL, MongoDB, or PostgreSQL, to store the data.

Implement CRUD operations

3. Next, you need to implement CRUD (Create, Read, Update, and Delete) operations for each of the entities defined in step 1. This will enable you to add new students, semesters, subjects, and marks, as well as update or delete existing data.
4. Implement the Three Reports
5. Finally, you need to implement the three reports as mentioned in the problem statement:

- Average Percentage of the whole class in the recent semester: To calculate this, you need to fetch the marks obtained by all students in the most recent semester, and then calculate the average percentage of the class.
- Average marks of students in a subject: To calculate this, you need to fetch the marks obtained by all students in a specific subject, and then calculate the average marks.
- Top 2 consistent students across all semesters: To calculate this, you need to fetch the marks obtained by each student in all semesters, calculate the average marks obtained by each student across all semesters, and then return the top two students with the highest average marks.

6. Implement Web Application
5. Finally, you can use Spring Boot with JSP to implement a web application that provides a user interface for adding new data and viewing the reports. You can also use Elastic and Kibana to visualize the data and generate reports.

Note: This is just an outline of how to approach the problem, and you will need to do a lot more work to implement this system. You may need to learn and understand several programming languages, libraries, and frameworks, such as C++, SQL, Spring Boot, and JSP