Design Document:

**Title : CSV Roster Application**

**Overview**

The CSV Roster Application is developed to import a classroom roster file, provided via a URL, into a relational database. The database should contain at least three tables:

* student (information about students)
* teacher (information about teachers)
* enrollment (information about which students are in which teachers' classes)

**Context**

There are 4 types of CSV files handled in this application

1. straightforward CSV file which has the correct data to be updated in the database.

2. A CSV file that is has larger student IDs, and exhibits a problem with long numbers that is often seen when the CSV file has made a trip through Microsoft Excel. It contains at least one record that will be rejected by the program.

3. A CSV file that has characters not present in the normal ASCII character set.

4. A delta roster file with a leading column titled "Action", containing either a "A" for 'Add' or a "D" for 'Delete'. Other than the addition of that column, the file has the same structure as the others. The listed students should be either added or removed as specified in the initial column.

**Goals**

This Application will take the URL of a CSV file as a command-line argument, along with a flag to indicate whether this is a Delta import(Y/N). Then the following operations will be performed.

1. connect to the CSV file's URL
2. read the CSV file
3. insert records into the database
4. report how many records were modified
5. report a list of each teacher in the database, and a count of how many students are in a class held by that teacher
6. and then exit.
7. In addition - if any records are malformed, the program will provide an exception log of any records that it rejected.

**Technical Details**

This application is developed in Spring Boot with Maven for build.

Language and JDK : Java 8

Database: MySQL

CSV Reader: Apache Commons CSV library

JDBC: For Database Connectivity

**Assumptions**

* The URL for CSV file should be public. If the CSV resides in a private URL or resides in a local directory, the file will not be accessed through this application.
* The row of records will be rejected only if ASCII values are present in the first name or last of the students. The application will also reject records having longer student ID’s.

**Other Design Ideas that were Considered and Rejected**

Using a JSP for front end (i.e, to get the file name in a text box instead of a command line argument). The reason for considering this approach is to make the project more presentable to the user.

Reason for Rejection:

Though this approach was does not require any command line to provide the input, it will give not give a good looking output, since, there are lot a values to be displayed as the output. Also, for huge number of records, the user should stay in the same page without refreshing or hitting the back button. If by any chance, the user closes the screen, the session will ended and the program will be exited.

**Questions:**

* How would you change your program if it had to process roster files of over 100K records or 1 million records?

BatchUpdate can be used instead of looping and updating every record. Through Batchupdate, all values of all the 100K records will be sent at once, instead of sending each and every record row by row.