Iteration 3 Refactoring document:
Overall purpose for this document is to address code smells and design issues that we have fixed/improved in our student management system before releasing the product to our customer. Our goal is to provide the customer with a program that shows good design principles and minimizes the code smells so it's easier for them to understand.

Refactoring done on Code Smells (problems and solutions):

Long classes:
Initially, we had large classes in the frontend and database that could be refactored. And some classes were short. So we safely extracted some methods and looked for common behavior within the class and added those methods to shorter classes without affecting the code. This helped reduce the size of the class and the readability significantly. Some classes will remain long as they weren't able to be shortened due to the structure of the classes, although the layout is in a readable manner.

Commenting on complex code:
We noticed before this iteration there was a lack of commenting so it was hard to understand some parts of the code in general due to its complexity. We have added a lot of comments all throughout the code that can clearly state what each method is doing so that the customer is able to read the code. Now after the refactoring process, the quality aspect of the code has significantly increased and the code is easy to navigate for anyone. .

Renaming Variable Names:
The variables names that were initialized were too vague for our system. When we were going through the code and making changes, we got confused over the variable names. We decided to refactor them since it was the root cause of most confusions. For example, when we initialized a student name, we named the variable name initially. This brought confusion as this could be admin name, student name, or even faculty name. So we have gone through the code and fixed all the generalized variable names to something meaningful like studentName and studentAddress. This is just one example provided, we have renamed many other variables to an appropriate name. Now after the refactoring process, the variables are meaningful which can increase the readability and understanding of our software.

Design Issues:
Exception Handling:
There were some methods that contained poor exception handling which may affect the software in the long run, so we have refactored some exceptions that were most suitable for the given case. For example, for the database we have added try catch blocks for the methods in case something goes wrong.In the database, the method catches SQLException and prints the stack trace using e.printStackTree() method. Printing out the stack trace is helpful for debugging purposes as it provides information about the cause and where it occurred in the code. Now using these specific exceptions in our code, this will help future issues that occur with the customer to be identified and resolved by developers.