-Error checking was not focussed on. This is particularly noticable for many ranged input prompts (i.e., “choose an id to edit”), where inputting an out of range, non existent id, or incorrect type of data, for example, string when expecting an integer, will most likely crash the program. Since input verification was not a requirement, it is recommended not to give the program invalid input.

-only one rubric per degree, but this is stated in the project description

-not really consistent when giving user choices: sometimes it is the id of the item, sometimes it is presented in numeric order

-no search ability. this was not required.

-a lot of sub menus to get to information

-when editing a field, have to reenter information, cant just edit it in place

-no ability to cancel an operation, but this shouldnt really matter because its ui related

-when something is "selected", it is repeatedly selected from the database to ensure details of the select object are kept up to date. this could be seen as inefficient, as i could have implemented some system to update the selected object in memory in addition to on disk ect ect

-educations are related to applications, and there can be unique educations for each application per student. this is designed to fit requirements, which desired that educations be a part of an application. however, in real life it would make more sense to attach the education to a particular student, and that student would be then be able to choose which educations they wanted to be present on any given application. however, if it were designed this way, the menu for creating educations would make most sense to exist in the menu for editting a student, and not in the menu for editing applications. This makes it seem like the requirement of educations being a part of an application is not being fulfilled, and so it was decided that the requirements would just be followed, and so educations were added to the application editting process

-As it was stated in the requirements that degree name would be unique, it seemed to heavily imply that we should use the degree name as a primary key, and so I did decide to do that. This saves an additional table column and computation time to keep track of a degree id. However, since the degree name is editable, this implicated an ON UPDATE CASCADE constraint on all tables referencing this degree name. Although I feel like this trade off is negative, I decided to continue using the degree name as a primary key, as it seemed the requirements would want it that way.

-there’s no confirmation on deletes

-there are a lot of menus to navigate through, and might not be immediately clear to a new user where they should go to be able to create or edit something, given that this new user knows nothing of the structure of the system

-the essay can’t be editted in a traditional sense, when you are able to navigate the original text and make modifications, you can only reinput a whole new essay which updates the field. But it wasn’t really in the requirements that we must be able to directly edit the essay.

-Inconsistent UI choices. I didn’t really pay attention to if the choices started at 0 or 1, and it kind of randomly switches around. Additionally, sometimes each choice is given a relative number, and sometimes each choice is the id of the choice, if applicable. Please be careful selecting choices – I don’t think I wrote any confirmation message prompts.

-The character limits for the varchar attributes in the database might be considered rather small.

-No value limitations outside of those required. Although there are some values that could have been limited, for example, keeping GPA between 0 and 4, this is not done as it was not an explicit requirement. However, relationship constraints between the main required entities are in place, for example, a student cannot create an application for a degree that does not exist

-Cascade deletions. There were not requirements on what should happen should deletion events occur, and so I kept it simple to conserve time, and have deletions cascade. Please note that additional data may be lost when deleting something. For example, deleting a degree requirement will delete all the answers to that question. This is in place so a student cannot answer a question that does not exist. Adding the question again will not recover the answers.

-At first I thought of saving the table columns somewhere in the program so it doesn’t have to keep figuring them out each time the user wants to edit some column, but that was a bit too much thinking, and apparently we are allowed to have a pretty bad interface application, so I stopped worrying about it

-no "best to worst" ordering for criteria. This isn't a requirement, but it would make sense for the possible scores to be ordered in some way. However, since it was not required, the database has no support for reordering of answers, and since answer scores are text/description based (e.g., "very good" or "poor" instead of a number), there can be no automatic sorting

-justification for multiple education redundancy: it is implied that the information for education is typed in everytime a user makes a new application on the user side interface website or something. So it is not the database’s job to make sure it is something that was already entered?