For our project, we listed the tasks that we have to do in another website named Todoist, and as a pair, worked down the list whenever we have the free time to do so. As such, it is frequent that when one of us does the backend work, the other would continue with the front-end work and fix all the bugs that he faced. For me, the work that I had done is usually the validation of user input and storing it into the database system; for example the LoginForm on our index page, the validation of the TagForm in the tagging of the questions, and the storing of annotation data from the user. Validation of user input also includes the validation of urls entered by the user, and ensuring that they do not pass invalid inputs in a ‘GET’ request. What I had learnt, and overcome, is that validation of inputs can be quite tricky at times, as we need to think of alternative ways that a user can crash our program (whether unintentionally or intentionally). For example, to validate the tagging of an answer, we had to check whether the question the answer is tagging with exists, so as to link the question and answer together, while ensuring that no two questions have the same answer. All these has to be done with the possibility of the user leaving out certain fields and filling some only, making it necessary to check for null values. This makes validation difficult. Moreover, I also learnt that sometimes we have to “bend the rules” so that our program can fit its purpose. For example, in an ideal 3NF database, there should be no transitive dependence; however, we know that the user may be lazy tagging every single image and wanted to do only part of it (for example, only tagging the question paper title for all the questions and answers). Instead of throwing an error and giving users a hard time to fill up the tagging form, we just decided to store whatever we have in a database so that user data would not be lost. However, that may mean that the values for the tagging of answers may be incomplete to correctly identify the corresponding question, so we decided to break 3NF and store all the corresponding question details the user gave into the database, and only updating QuestionID in the Answers table when there is enough information to correctly determine the question that is linked. Hence, sometimes it may be better to bend the rules a little; and then come back to fix it once we have the idea to refactor it.

For me, the most proud implementation that I had done is all the bug fixes that I had done in the entire project, especially the fix for the https vs http issue. Our program automatically blocks requests in http if the website is in https; however redirects may sometimes redirect a https page to a http page; hence resulting in an unresponsive website. After searching long hours for a fix, trying meta tags, SQLify, Talisman (which caused infinite redirects for no particular reason), I finally fixed the problem by adding \_scheme=”https” and \_external=True to url\_for, and adding meta tags to upgrade insecure requests to htts. However, that would not work for localhost (since https isn’t supported in localhost), so I searched and decided to redefine url\_for to auto strip of \_scheme and \_external if it passed to it while being in localhost. Bugs (especially tricky ones) are magical - facing it makes a programmers’ morale plunge, fixing it boosts a programmer’s morale so much, it makes him feel special about himself for making things finally work. And this is how I felt when squashing computer bugs.