Project 4 Report

and Expectations for Project 5

Project group members:

Joe Spahn

Aryan Jain

Jessica Dresmann

Jonathon Schroeder

PART 1

Our first step to completing this project, after reading the handout thoroughly, was writing down a list of all the classes and methods we knew that we'd need from the beginning. These included classes for teachers, students, courses, quizzes, and more. At the same time, we also agreed that the best two options for our selected features would be randomization and grading, as they seemed to make for the best LMS tool, and file imports didn't seem very user-friendly. Unfortunately, we never implemented any optional features, as we spent the whole period for the project perfecting the program.

The requirements for login and account creation stated that users must be able to log in to previously created accounts, even if the program has stopped since the account was made. We are using file input/output in order to write accounts down onto .txt files. Teachers and students have separate files we are reading from and writing to, and we require usernames to be unique for each account, as they will be used to identify each individual student's scores. In each login class, we have methods for adding, deleting, and editing accounts. Whenever one of those methods is used, it will use a separate method to update the text file. There is an additional method used when creating accounts that checks to make sure a username has not been used before, which implements the method that reads through the files. Then, when a user logs in, that same method that reads through the .txt files will compare the username and password. If it finds a match, that user will be logged in and given a new menu based on whether they logged in as a teacher or as a student.

A teacher is given the option to choose a course or create a course. A student is given the option to take a quiz or view their own submissions for a quiz. Both users can also edit their

passwords or delete their accounts. Much of the course creation is similar to the account creation in terms of file i/o, but quizzes were much more complicated. After creating and accessing a course, a teacher is given the option to make, edit, delete, or view submissions for a quiz. There was a great amount of information that we needed to write down for each quiz: The course it was a part of, the name of the quiz, whether or not the quiz was to be randomized, the name of each question, each question's options, which answer was correct, and how many points a question could be worth. Both of our selected requirements were implemented here, the randomization of the quizzes and the detailed score reports for each quiz. We ended up using separate classes and .txt files for questions and quizzes. When a student takes a quiz, the program checks for a quiz with the name the student gives from Quiz.txt and displays questions that it reads from the Question.txt file, then it records both the total score of the quiz as it continues as well as the score a student got for each question. That way, when a teacher or student views that quiz submission, they can see what questions they got right and what questions they got wrong, as well as their total score, and what time the student submitted a quiz, in case it was late.

The program reads and writes information down from various .txt files constantly in order to maintain attributes even when the program ends. The classes used in this project are all interconnected, calling on one another to create an experience that seems smooth and simple, but on the inside is rather complex and sophisticated.

PART 2

Joe Spahn: I contributed to the project by making the courses, quizzes, and questions classes. The course class is the class that is created by teachers and can be accessed by students. This class holds all the quiz objects for the class. The quiz is a list of question objects. This also is where the methods for adding and deleting questions are. The question takes a question string, an answer string, and a list of options. The constructor adds the answer to one of the options this class also randomizes options. I also coded the teacher's side of the main menu. I also contributed to the design of the project and how the project was going to flow. I also helped in working on the connectivity of the project. This was to make sure that even though we were all working on different parts and classes our program still had to work together. If one part was out of place and wasn't right the whole program would be ruined. So it's very important that all the method names were constant and that all the classes would work together. So I communicated with my team to make sure this was the case and explain to them any issues I saw or any questions or comments I had.

If I could do anything differently I would have worked out the IO part of the project first because that is what took the most time and ended up being the most complicated. We finished other classes that required the IO to work. We could not test these classes until the IO was done. For project 5 I think to avoid this we start with the most complicated part first and the one that is most connected to the rest of the program. This would allow us to test our methods and classes as we work instead of having to completely redo them to accommodate the IO.

Aryan Jain: I contributed to this project by creating all the student and teacher classes, the methods for which are accessed whenever a student or teacher chooses an option from the menu. A large part of these classes was creating the methods that were used in quiz-taking and viewing submissions. When a quiz is accessed, it first checks to see if the questions should be randomized and then returns a list of questions that will be shown to the student. An especially complicated issue with quiz taking and submission viewing was finding a way to score every question individually, so I made a new class called ScoreQuestion which is implemented in the main method every time a question is answered by a student and creates an array list for each correct and incorrect answer. Once a quiz is finished, I created the method that submits the quiz to a .txt file with the total score, the score for each question, and the time a student submits the quiz.

I also made sure that everyone in the group was staying on task. It was a very difficult project and there were many days we were ready to slack off and avoid it, but I made sure that we kept on meeting daily in order to come out with a great final project. Furthermore, as the project got more complicated and the roles we took on initially started to become less clear, I helped make sure everyone knew what they were supposed to be doing and every team member was working on what they did best, I helped everyone in debugging and helping them understand a concept if they faced a problem implementing it.

If I could do something about Project 4 differently, it would be to get a clear picture of everyone's schedules so that we can regulate meeting times and be most efficient with balancing this project with our other responsibilities. My team members still met up nearly daily, but it was sometimes difficult to work around exams and other big events. For Project 5, I will discuss with my group what times are best for us on the week and weekend, as well as when any of us have extra time to work on the project or no time at all.

Jessica Dresmann: My greatest responsibility for this project was the classes and methods for teacher and student account creation and login, but I ended up working diligently on many different classes and implementations once I was done with the accounts. The login system was a heavy burden as this was the first part of the project we would have issues with, and it would need to be finished before we can start debugging further parts of the project. The file input/output was a little daunting, but in the end, I came up with classes and methods and wrote down usernames and passwords that could be accessed in the login. The usernames were made to be unique because later on in the program usernames were used to identify individual students rather than passwords. Once I finished the account creations, the login came easier, and my main role was finished. In order to keep contributing, I focused on debugging, locating errors, and fixing them. I worked together with Aryan Jain on all the login classes themselves, where viewing quiz submissions was becoming increasingly more frustrating. Finally, while the project was being finished up, I made many of the test cases, which took much longer than I had expected.

If I could do anything different for this project it would be to include more self-teaching and utilizing our Purdue computer science resources. We went to the USB help room and talked with some TA's, but most of this project we did on our own through brute force. When we begin Project 5, I will make sure we all know what topics we should be researching in our free time to become more familiar with them, and I will also encourage my group to use our TA office hours so that we do not feel like we're doing this all alone.

Jonathon Schroeder: My role throughout this entire project was clear, the main method was my responsibility. I took that role happily as I knew I could rely on my team members for help whenever I was implementing their methods in the menu. There were lots of errors in the menu throughout the project, but I did my best to weed them out and fix them. One of the greatest issues I faced time and time again was making sure that the system could take invalid inputs and repeat prompts in those cases. A large portion of the menu, the options for a teacher after they logged in, was actually done by Joe Spahn while I was working out issues with the login system. It was very satisfying creating the menu, as I could always test it immediately when I needed to, and get rid of many of the bugs before anyone else noticed them.

I also tried to make sure the group stayed positive and didn't give up. I felt lucky to have a group that was so willing to meet and work together, so I wanted to make sure that we didn't take out our frustrations from the project on each other. In the end, that never was an issue, and we all kept our heads up even when our errors seemed unfixable.

If I could do anything differently about this project it would be to communicate more with my team members about the work that they had done. Since I was the one writing the main method, I should have talked with them more about their methods so I could implement them properly, which I did start doing more as the project was coming to an end. For project 5, regardless of what I end up working on, I will make sure to talk with my group members more about their work so I have a greater understanding of the program as a whole.

PART 3

Our collaboration strategy for Project 5 will be similar to our collaboration strategy for Project 4. We will have a discussion when the project begins about what classes and methods we know we will need to create, and about major themes in the project that will come up, (concurrency, GUI creation, Network I/O, etc.,) After we have detailed and written down the different aspects of and classes within the project, we will give out roles and decide who is working on what. We should know from prior research, experience, and how we all performed in project 4 where our strengths lie and decide our individual responsibilities based on that. Our final step before beginning the project itself is coming up with a schedule of when our best times to meet would be, and our deadlines for certain parts of the project. Those can be written down in a separate online document for reference when we meet. Once the project itself begins we can start another document where we detail our errors or ideas. Our group also has a chat room and we will continue to use that to discuss meeting times or important information, such as commits that have been made or describing work done in a member's free time.

Our main goals for Project 5 should be to finish concurrency, Network IO, GUI implementation, and file import/outport. Making sure we understand GUI implementation soon is important for testing and debugging so that should be done first by November 24th, and Jonathon would be best off working on this. File import is something that Joe has proved he excels at and he will be expected to have it finished by November 28th. Network IO and concurrency are two of the most difficult aspects of the project, but Jessica and Aryan seem up to the challenge. They will be used in many of the same places, so they will be due last and at the

same time, on December 6th. This gives exactly a week for test cases and making the report for the project to easily be finished by December 13th.

While our group doesn't seem prone to arguments or conflict, it's important we have a strategy to avoid and overcome it. If conflicts about the code itself arise during Project 5, we should take a step back and discuss it as a group, or even talk to a more experienced TA so that we all have the same idea of how the project should be done. If a team member begins to slack off or not show up during meetings, we will call them out on it and it will be reflected in their report at the end of the project. However, proper planning and scheduling of the project should quell any conflict before it even begins.