**Project 5 Report**

Andrew Wu, Jebran Syed, Arul Chauhan, Ryan Chen, Zach Garcia

**Part 1:**

The program that our group decided to program was option two, the learning management system quiz tool. At the beginning of our program, a graphical user interface will pop up and prompt the user to create an account or log into an existing account, not allowing a user to continue through the system if the account does not exist. The interface will continually update throughout the program based on the user’s choices with different prompts, questions, and buttons relating to the different functions of the program. When creating an account, the program will deny account usernames and prompt users again if they have been taken already. Once the user has a valid account that can access the system, the features the users can access are dependent on the type of account the user has created. The different options that a user may be able to select will be in the form of pressing buttons that appear on the interface, which correspond to the action written on the button. If the prompt is an open-ended question, users will be able to type out their response to the given prompt. If the user has created a student account, they are prompted with options to view quizzes or edit their account. If a student chooses to edit their account, they are able to edit their account credentials as well as delete their account if they wish to. If the student chooses to view their quizzes, they are able to take quizzes and view quiz scores based on the courses that they are enrolled in. If the user has created a teacher account, they are prompted with options to edit their account, create a course, edit a course, or view student scores. If a teacher chooses to edit their account, the teacher would be able to change their username, password, and delete their account, which is the same features as a student. If a teacher creates a course, they will be prompted to input a name for the course and the course would be created. If a teacher chooses to view a student’s score, they are prompted to choose a student and then they will be prompted to choose which quiz to view, and the score will be able to be viewed. If a teacher wants to edit a course, they would be able to create a quiz or view an existing quiz within the course. If the teacher chooses to create a quiz, the teacher can create a quiz by importing a file or use the interface to manually create a quiz. If the teacher wants to view an existing quiz, they are then prompted again and can choose to delete the quiz, view student scores associated with the quiz, or edit the quiz. If the teacher wants to edit the quiz, they can change the name of the quiz, edit existing questions or add questions to the quiz. We also added ways to exit out of prompts for both teacher and student features, so that there is a way to go back to the main page if a user wishes to do so. With the way we organized our program, we allowed the program output user interfaces that the user can interact with to efficiently execute the user’s intentions.

All these features within our program work at once, meaning that if a teacher were to make a change to a quiz, the change would be instantly updated on the student’s side of the program. Likewise, if a student were to take a quiz and complete the quiz, the scores would be instantly updated on the teacher’s side of the program. This is made possible by implementing concurrency within portions of our program, which allow information changes like those highlighted before to be instantly updated for other users currently accessing the program. Furthermore, all information, regarding accounts, courses, quizzes and related scores, are all stored within a server that a user must connect to through a client. The server acts like the cloud in a way, where information stored on the server would be updated instantly and used by other clients instantly, instead of information being stored locally on the client side. By using the server to store information, it allows multiple users to access the same, updated, information at the same time. It also allows information to be stored elsewhere should the local client have any issues, causing the connection to the server to disconnect and local information to possibly be wiped. Multiple users are able to access the program as well. Multiple students and the teacher of the course would be able to access the program and its features at the same time by the use of threads within our program, which is a feature of our program. Through the use of servers within our program, threads are created for each user that is accessing the server. The creation of multiple threads thus allows multiple users to access the program at the same time, and through concurrency within our program, the information displayed on the interfaces that each user is looking at can be updated with new information once the information has been created. The features we included within our program were organized by who can access them and by what each feature accomplishes. An example would be a teacher editing a course. Both creating and viewing quizzes were placed within the edit course option, due to the similar actions that they execute. Within our program, teacher and student accounts have different features that are accessible to those account types. For example, only student accounts are allowed to take quizzes and only teacher accounts are allowed to create and edit those quizzes. By organizing our program by access of features and organizing them, our program is able to divide features by who can access them and organize them properly based on similar characteristics. By implementing all of these features, our group has created a program that is able to meet the required features of the consumer.

**Part 2:**

**-Andrew:**

Within project 5, I helped out with creating the GUIs and took care of most of the non-coding aspects of the project, including writing the report and creating the presentation for the group to present with. I specifically helped structure the main menu GUIs and creating the interfaces for the overarching menu area, including the welcome GUIs, the main interface GUIs for student and teachers, and more. With the report, I wrote up all of part one, consisting of writing about what our program does, the intricacies/details of the program itself, and the features that the program has. I also wrote about things that we changed/added in project 5 from project 4. I also created the presentation that was used within the business pitch that was presented. The main focus of the presentation was to highlight the features of the program and use the features as the main selling point for the program. I also included a few slides to pitch the program that allowed us as a group to persuade the audience why our program is better than other programs and systems currently in the market. The slides used to pitch our program were also used to persuade the audience why consumers should use or switch over to our program instead. The presentation finished off with a few questions, which would allow our group to clarify some specifics about our program that we did not cover within the initial pitch.

One thing that I think I would change if I had some time to do so would be to spend more time on the GUIs to try and make them look better. Currently as it stands, the GUIs are at the bare minimum of what they can be. They work correctly and are able to go through scenarios that the user may ask the program to go through, however, they are the bare bones of what the GUI can be, an interface with some text and some buttons with the occasional typing bar for typing answers. I would have tried to make the GUIs that I worked on more organized instead of just having a bunch of buttons. Some ideas that I had were to add symbols either on or next to buttons to add imagery to what each button would do. For example, I would add the typical gears icon used commonly as the settings symbol to the edit account button within both teacher and student features.

**-Ryan:**

Upon our first group meeting, I was assigned along with Arul to complete the concurrency and threading components once the GUI was finished. Once concurrency was completed and set up utilizing the classes Server and ClientThread, I helped with rehauling parts of the Teacher class-related GUI such as the methods in menu titled randomizeQuizMenu() and attempted to complete importQuizMenu() whose functionality was to take a file path from the OS system and a quiz name entered by the user and to pass it to the ClientThread class for additional processing. Additionally, I helped make the GUI pathing starting from the edit course menu. From the createCourseMenu() which shows a drop down of possible courses to choose from, I implemented a path that connects that menu to the main editCourseMenu() which displays button such as edit course name. Additionally, the main editCourseMenu() also displays a list of quiz names under the course name with it as the JFrame title. At the south layout of frame, I added three buttons with the functionality create quiz, which directs the GUI to a createQuizMenu(), an edit/delete quiz button, and a view quiz button. In the createQuizMenu(), I added additional functionality that leads to a createManualQuizMenu button, a randomizeQuizMenu() and an importQuizMenu. I additionally helped with the pitch of our project and some limited testing.

If we could do this project over again, I would have advocated a more in –depth planning period. Additionally, it would have been better to not try to adapt project 4 over directly. Instead, it would have been better to plan an overall rehaul of project five and include all the functionality needed. In regards to adapting over project 4 methods, we could keep the underlying logic of utilizing file systems to store information but we should have thought about adapting the methods and classes from project 4 instead of basically copying it and trying to format GUI around it. Additionally, since we did end up using a file-based system from our project and for information permanence and storage, it would have been smart to implement a file manager tool that comes already with the GUI swing package, this would have made the flow much easier. Lastly, it would have been nice to add more pictures and a better-looking GUI for user experience.

**-Arul:**

I helped Ryan with the concurrency features to make sure that data was updated in real time. This meant that students and teachers could communicate by going back on pages and the data would be updated the next time the user enters that said page, whether it be an updated course or a updated quiz. A lot of users will also be able to connect to the server at the same time, since each time a user connects to a server a new ClientThread is created to handle their requests. This thread is unique for each user, and has a unique username and text file associated every time the user logs in. I also created some of the student functionality, where they are able to view courses, view the quizzes in those courses, and take a quiz. After taking a quiz, the student’s submission file will be uploaded to the server, where the teacher can then view and grade the quiz. This grade will then be written to the file, where the student can view their grade given by the teacher whenever they want. The submission is also dated with the current date and time, and the student is allowed to make multiple submissions, which can all be dated and graded.

If we started over, I would really like to look more at the GUI and make it look presentable instead of a bunch of buttons put in a grid. Although most of the functionality is working, the presentation of it is quite lackluster and the user might be a little confused based on what the GUI looks like. Another thing I’d like to do is improve our database system. As of now, we are creating separate user files with identifiers based on whether they are a teacher or student. I’d like the system to be more robust, with hashing for the password field if someone manages to get into the file system in the server. Lastly, I’d like there to be more unique functionalities like the teacher being able to give more time to the students and specific requirements for specific students, so that the platform would better act as a finished product.

**-Zach:**

I was primarily responsible for the server class and client interactions but I also helped with the design of certain features in the teacher flow. Specifically, I created menu methods for editing a course’s name and adding a manually created quiz. Each menu method subsequentially calls a new menu method for the next possible stage in the program. I was also responsible for editing and deleting quizzes as well as deleting courses. The edit course path allows the teacher to edit the name of the course, along with creating quizzes, importing quizzes from files, editing anything within those quizzes, and deleting the quizzes as well. Teachers may view submissions and grade students’ answers as well.

Given a second opportunity for this project, I believe we should have given ourselves more time. We also greatly underestimated how much of the project was involved with GUI and overestimated how much of the project was concurrency and server/client. This led us to distribute two individuals to GUI, two individuals to concurrency and one individual to server/client. The concurrency and server/client aspects were finished early and the GUI aspect of the project remained unfinished.

**-Jebran:**

For project 5, I was primarily involved in identifying areas where the GUI was required and more importantly what type of GUI was required. For example, in the create account and login windows, simple GUI would not work as well as complex GUI. For windows which require users to select what they want to do and required various components all in one place, complex GUI was required there as well to accommodate various buttons and labels. As a part of this process, I had to use learn to use various tools associated with Complex GUIs such as the .setBounds, Radio Buttons and Combo boxes. Using this, I converted a good amount of the System.out (print commands) and System.in (scanner commands) commands present in project 4 that took in user input in all of the classes in the program into GUI based commands that allowed for user input. Additionally, I also had to use simple GUI to display single question messages such as course names or quiz names as well as error messages that are to be displayed.

Since we had to use GUI, I also had to work on certain aspects of the flow of the program in the Menu class since it could not accommodate do-while loops inside action listeners. Furthermore, since the program had to flow based on whether or not buttons were clicked, this meant ensuring that the right questions and methods were called after clicking a particular button through the action listener.

If I were to do this project differently, I would work on making the GUI much more visually appealing to the user. While I was working on the menu part of the project I also realized that trying to build upon Project 4 may not have been the best approach to work on Project 5 given the presence of action listeners and the inability to use do-while loops to go back to main menus. This required a lot more planning than we had anticipated which we should’ve done earlier. Additionally, exploring Java’s swing further could’ve made our file lists more user-friendly for the user and a few other features of our code much more robust for the user.