Currently, the course grading application is working on the elements machine, relying on Perl scripts and written in PHP, we are working to turn it into a web application. Graders and teachers can login and grade assignments; however, it requires access to the University of Pittsburgh elements machine via an AFS login. The application is being redeveloped to be a web application, and it is our job to develop a database and integrate it into what is already built; along with integrating a secure login system into the application.

Students can upload their assignment files, and the auto grader runs and either accepts the output or denies it. When accepted, the student gets a sub-score, but before the score is finalized it must be checked by a grader. The grader or instructor will login separately and be able to look up sections to find students and assignments needing a final check. The grader will look through the source code to make sure that the student is not just using print statements to match the output and is doing their work before finalizing the grade. The database is going to contain complex relationships between graders, teachers, courses, and students. Ian Manners has more experience with databases, so he will play a bigger role in developing the database. To integrate what we have built we will be modifying the source directly, the source is held on the elements machine, so we will require AFS login capabilities, and we will use some sort of file software like WinSCP to directly access the source code, and run it through putty.

For the login page, users will enter their login, and we will check this against our database to see if the user exists; afterwards, if verified, the user will enter his password. Verification of the password will be done by code that will be given to us; we must integrate this code into the current system. As a team we will have weekly meetings with Hoffman and communicate with him via e-mail; to communicate with each other, we will text each other and use discord for easier conversations.

The current course grading application is written using only PHP. One of Hoffman’s plan’s is to make the course grading app into a web application. This means that we will also need to use some javascript and possibly some HTML to help with the web application. Hoffman was also clear that he wants a MySQL database that is capable of storing information about courses, students, teachers, and graders. In order to do this we will be using SQL using the Oracle SQL Developer Kit. These will be the languages that we will be using for a majority of the project. The Software Framework for this application is something that we will not have any control over. This is because important parts of the application are already developed so we will be adjusting to the framework that is already being used in the current application. Hoffman wants to keep the current application mostly the same just converted into a web application with some more extensive database and security features. The current application uses a simple web framework for the student assignment submissions and grading. We will be taking the current web framework and changing it into a web application framework. The differences will be that the students, teachers, and graders can all interact with the application to submit assignments, grade assignments, return feedback on assignments, and read the students submitted code to check for plagiarism and logical correctness. For the most part we will not be worrying about framework too much and will just be modifying the code directly.

PHP has built in support for SQL via MySQL functions which will ease integration of the database into the system. We will create a database subsystem to manage all interactions with the database and provide uniform access. Everything else will be integrated as pieces of functionality and added directly to the source code. We plan to test our code before integration to ensure that it is working, then when integrating; we will have to upload our code piece by piece to the elements machine, coalescing it with the code already written, and test each part individually; as well as, testing it all together.

Testing the code separately will be simple, we will use print statements to ensure that we are getting the correct results from each function, and tools such as Visual Studio Code can directly view databases, so we can check if the data is stored correctly. Testing the code after integrating will be more difficult; we will have to generate fake data and make sure every function we’ve implemented modifies the data in the correct way. We will need to use Putty or something similar to access the University elements machine; as well as, a program like WinSCP which provides easy to use GUI to transfer files. Other than that, we will need an IDE that supports writing in PHP; such as, visual studio code with a PHP extension.

There are a few possible issues that could come along with this project. For the database, there are multiple dependencies and variable relationships that we need to keep track of. If we mess up a dependency or a relationship between important variables, we could corrupt the data of the entire database. We will solve this issue by making sure that we select the right keys for the dependencies and relationships so that they are unique and fast. Finding the correct key can be difficult sometimes so we will need to map our dependencies beforehand to make sure that they are correct. Another issue that will arise from the database is the difficulties to integrate it into the current application. The application, in its current state, only accepts student submissions, tests the output of the program against the correct output and a few minutes later returns a grade. There is then a backdoor for the graders to view the code files that the students submitted. What we will need to do is add the database that we created to the current application so that the teachers and graders easily have access to all of the students grades and files and the students also have access to all of their professors, grades, and submitted assignments if they would like to return to them to study. We will also run into some issues with the login system. As of right now, the application does not use multi-factor authentication like Hoffman would like it to. This means that we will need to integrate a script that is already developed for the University’s multi-factor authentication system into an application that is already developed. These integration issues could cause problems since they were written by different developers, on different systems, using different frameworks. We will solve this problem by figuring out a way to scale the application correctly. The final issues we could run into is actually converting the current application into an interactive web app that all the students, professors, and graders in CS can access for their information. We will need to make the web app organized, secure, and easy to use. The hardest issue to solve with this integration will be separating the levels of access between different user types such as professors, students, graders, and administration. We will solve this issue by using Pitt logins as logins for our system and by correctly writing the database securely. As far as the system that our application will run on, it will run on any system with a browser and internet access. It is a web application so there will be a domain in which anyone can access the login site, while only people with a valid login will be able to access the database. Access to the database will depend on your user privileges.

In conclusion, a lot of useful skills will need to be learned or refreshed for the successful development of this application. We will be brushing up our skills on databases, authentication, web applications, and most importantly integration. We have already developed some of our user stories for the project and will be getting started with development on the application soon. We will be adding user stories as we go on and Hoffman’s vision of the project changes. Below are our user stories for the start of the application.

**User Stories**

Story 1: The user needs to be able to login to the system using their Pitt Username and Password

Story 2: We need the login to be secure

Story 3: We need a multi-factor authentication system

Story 4: We need to move away from AFS and convert it into a web application

Story 5: We need a database to store the information of instructors, students and graders

Story 6: We want complex relations between instructors, graders, and courses

Story 7: We want to integrate user access from the database into current application