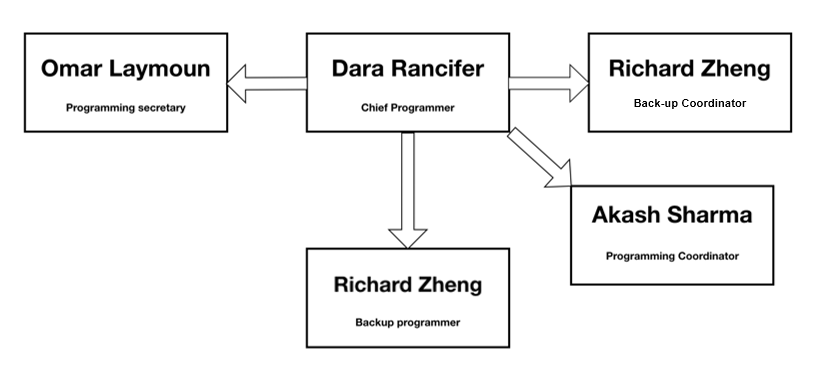
3321 Software Engineering

**Project Report**

**Team Members:**

Dara Rancifer , Akash Sharma , Omar Laymoun , Richard Zheng

**Team Model:**



*Chief Programmer Model*

This team model worked best to suit our needs as we all had different time constraints outside of the project. Dara had the role of the Chief Programmer and was solely responsible for programming the entirety of the software module, from the back-end to the front-end, and then the creation of the architecture and class diagram, and test cases. Akash took on the role of the Programming Coordinator, his job was to keep everyone in communication, handle meetings, and relay updates from the Chief Programmer to the Programming Secretary. Omar had the role of the Programming Secretary, he wrote and updated the Software Project Management Plan, and made sure the Chief Programmer continued to document the program. Richard was the Back-Up Coordinator and Back-Up Programmer. He relayed information from the Chief Programmer to the Programming Coordinator, assisted the Programming Coordinator in communication to the Programming Secretary. The Chief Programmer needed to keep the Back-Up Programmer updated always, in the event she was sick or unable to complete her duty. Richard also assisted the Programming Secretary by creating UML diagrams to help visualize the current and future status of the project.

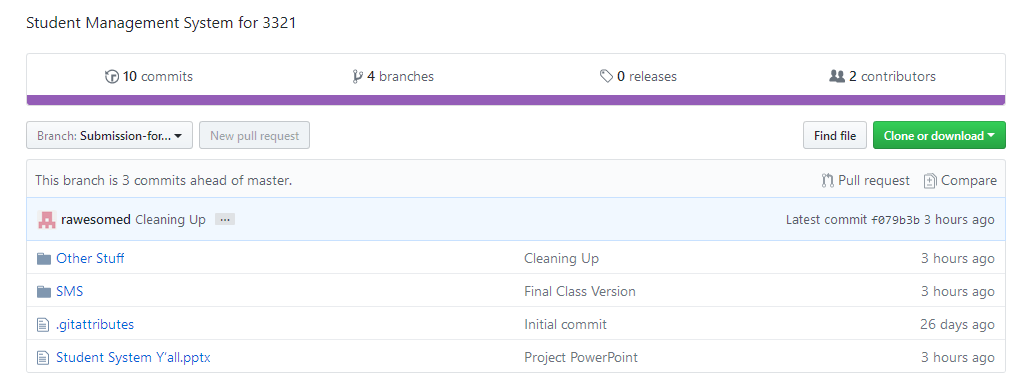
**GitHub:**

*Hosted by:* LunarSkyPriest – Akash Sharma

*Contributor:* rawesomed – Dara Rancifer

*Link:* <https://github.com/LunarSkyPriest/Software-Engineering-Student-Management-System-3321/tree/Submission-for-3321>

GitHub was our main form of Version Control. Instead of smaller, frequent updates. Every week a large update whether it be a functionality overhaul, or graphical update, would be pushed to Git. Akash, Richard, and Omar would view the GitHub from the browser to keep track of updates, whereas all commits were done using GitHub Desktop, for convenience. This kept in tune to the Programming Chief Model, since there was one programmer, the software commits were performed by one person.



**Software Project Management Plan:**

The Software Project Management Plan has been uploaded to the GitHub for reading and appending. It documents the entirety of the software project, from the requirements phase, to the final bit of testing.

**The Project Itself:**

The project was programmed in Visual Basic and compiled in Visual Studio. Despite my little knowledge of Visual Basic, it was still enjoyable to get a hands on experience with it, for the first time. Overall, it seemed like a success to use Visual Basic over other methods, due to the GUI being able to directly connect to the database table objects. To accomplish this with no prior experience with Visual Basic, it was important to create architecture and use case diagrams, in order to help visualize the necessary forms to complete the requirements. Knowing what I wanted to do, assisted in programming the back-end logically.

**UML Diagrams:**

*Below is the architecture diagram, class diagram, requirements and analysis use case diagram, implementation and design use case diagram, and test case models.*

**Architecture Diagram:**

**Splash**

**Login Screen**

**Student Side**

**Student Hub**

**Student Profile**

**GPA Calculator**

**View Courses**

**View Grades**

**Change Password**

**View About Page**

**Course Descriptions**

**Logout**

**Course Descriptions**

**View Grades**

**Admin Side**

**Admin Hub**

**Change Password**

**View About Page**

**Add Student**

**Edit Student**

**Add Course**

**Add Grade**

**View Grades**

**Add Grade**

**Delete Grade**

**Edit Grade**

**View Students**

**Add Student**

**Edit Student**

**Delete Student**

**View Courses**

**Add Course**

**Delete Course**

**Course Descriptions**

**Logout**

**Info Database Tables**

**User Database**

**Students Database**

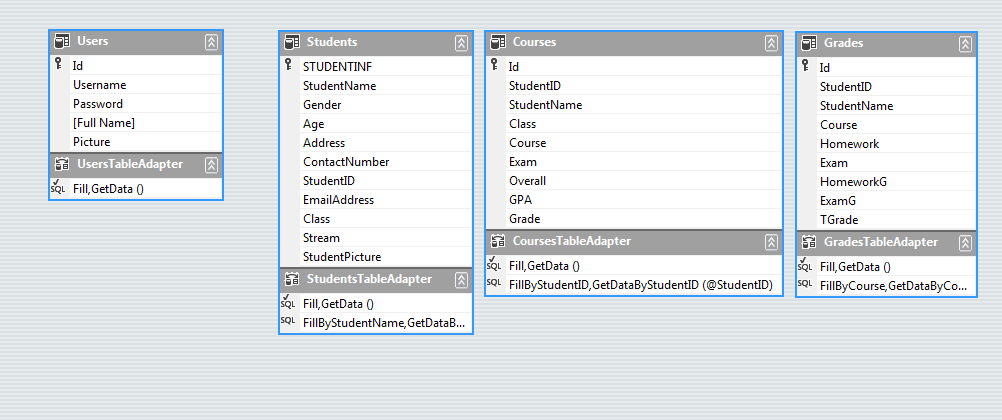
**Courses Database**

**Grades Database**

***Student System Y’all***

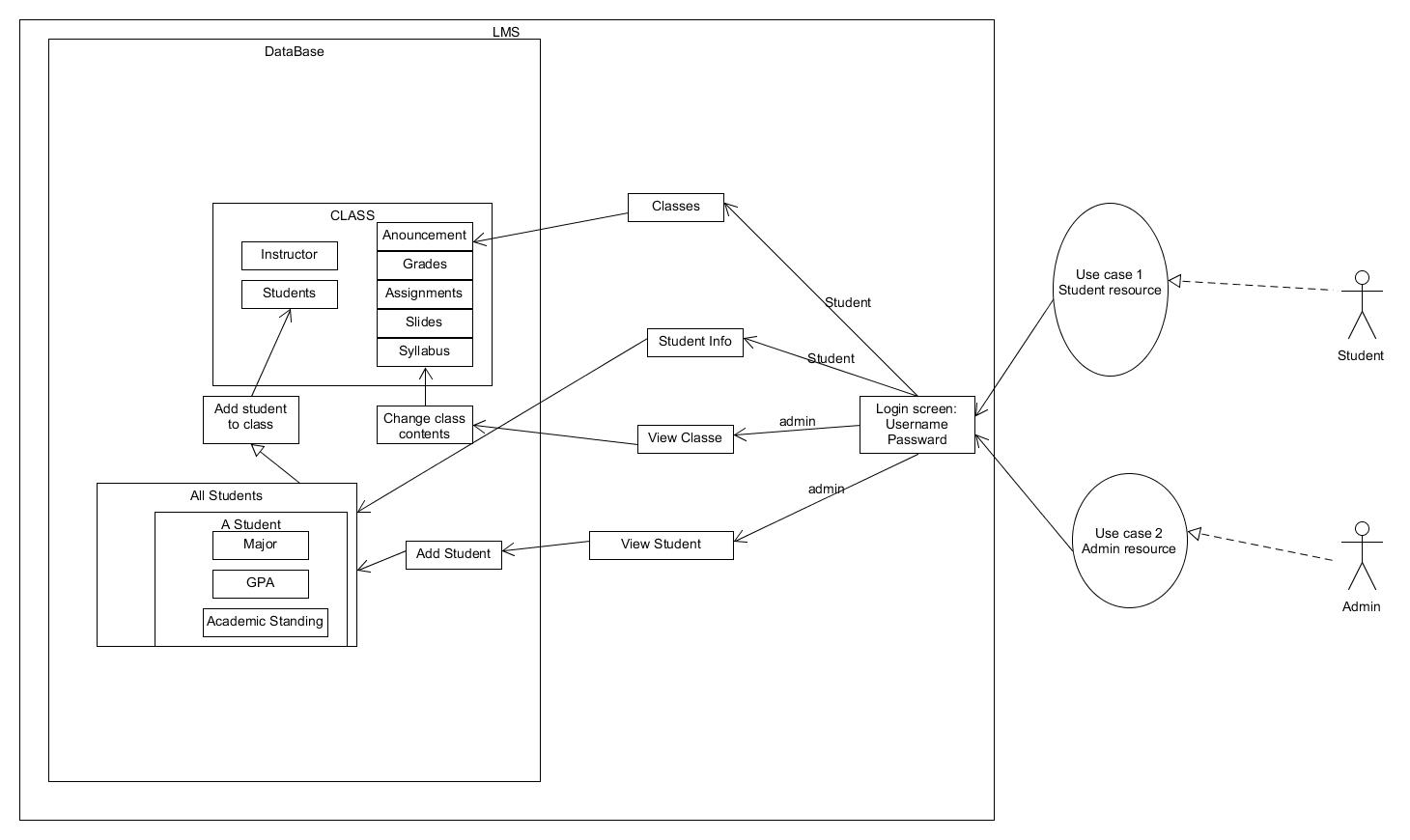
**Class Diagram:**

*How the classes primarily connect to each other, specifically in terms of the database tables.*

****

**UML Diagram:**

**Requirement and Analysis:**

*How we interpreted the requirements of the project and analyzed the best way to accomplish them.*

**Use Case Diagram:**

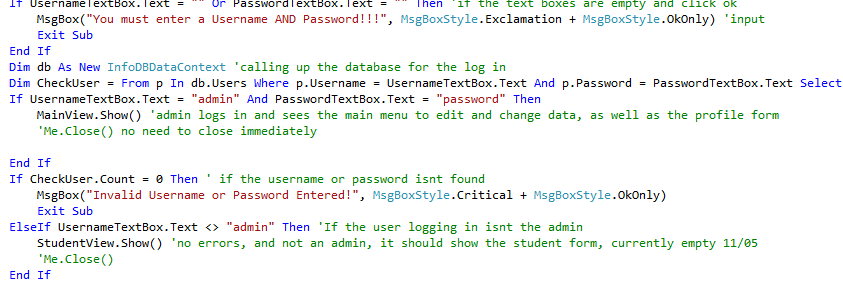
**Design and Implementation:**

*How we ended up designing the software module, and its current implementation.*

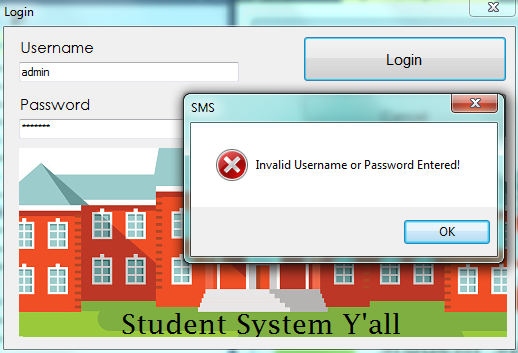
**Test Case Model:**

*To assure that the student and admin abilities were to not overlap, the split occurs at the log in screen. The student logs in with the corresponding student username and password. Upon log in, only the student will see the student hub. Whereas, when the admin logs in, they see the admin hub, which has access to the student, courses, and grades dataset tables. To guarantee no outside actors impact the system, error messages will be shown for the wrong information, a warning is shown when no log in information is inputted, and lastly, the log in fields are cleared after the admin or student has logged in.*

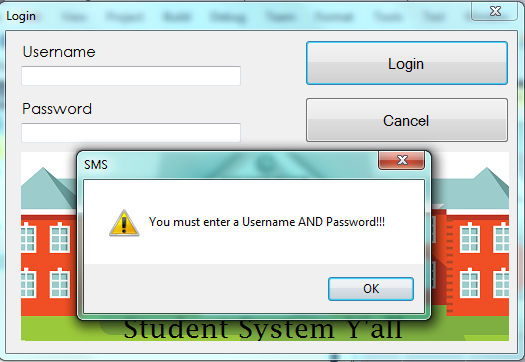
Splitting log in fields:



Wrong username or password (the data was not found in the database, despite the user attempting to call themselves admin):

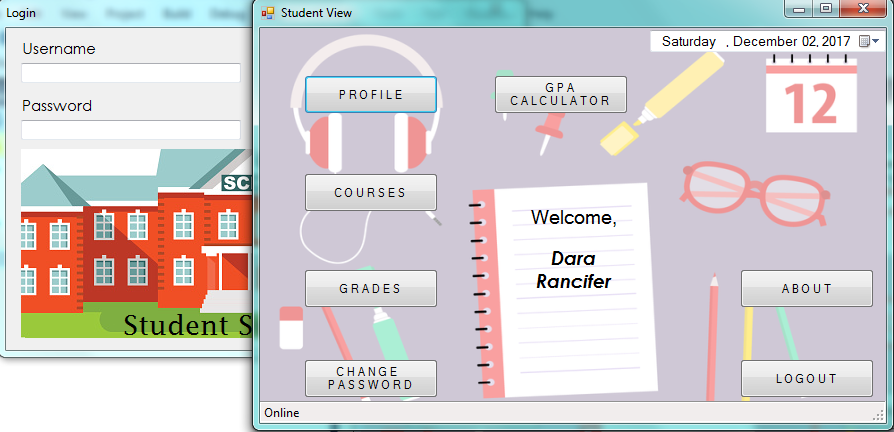


No username or password entered:

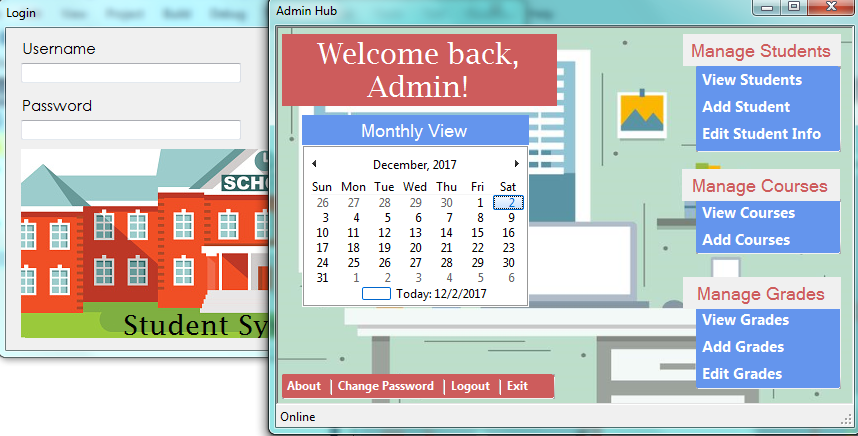


Student Logging In: (Note, see how the username and password fields will clear for safety),

*The student will even see their corresponding name to their log in information.*



Admin Logging In: (Here, the username and password fields will also clear upon login, for safety)



**Final Thoughts:**

Due to the events occurring this semester, we were all under a time crunch to not only complete this project, but others in different classes as well. In terms of what functionalities the software was able to complete, we are satisfied with how it turned out.

From this project, we learned that the Software Engineering process is not as simple as we once believed. There is a lot of factors that have to be considered before even entering the design and implementation phases. A lot of our time was spent deciding who would do what and how to interpret the requirements. There was a lot of time spent discussing the diagrams and what language to even program the software module in. In the end we decided that the Chief Programmer model would work since we just needed to begin working.

As the chief programmer, it was not easy. Hours of personal time were spent learning Visual Basic and implementing forms, function by function. Some nights a lot would be implemented, and others very little. Because of this, whenever key functionality was added, I needed to be able to share this information with the coordinator and back-up because they needed to adjust design models or share information with the secretary. It was even necessary to continually demonstrate and talk about the code, so that in the event I was sick, Richard needed to be able to take over. So for them, it was difficult too.

All-in-all, we have a new found respect for what it takes to be a software developer, engineer, debugger, and all the sort. For an undergraduate project, it was very worthwhile and truly explained at the end, why the textbook includes as much as it does. It is a lengthy process with a lot of steps because that is how it is in the real world.