**Software Engineering Documentation**

**Carlos Rios, Ivan Sosa, Satish Subramanian, Phillipe Hehn, Derek Jones**

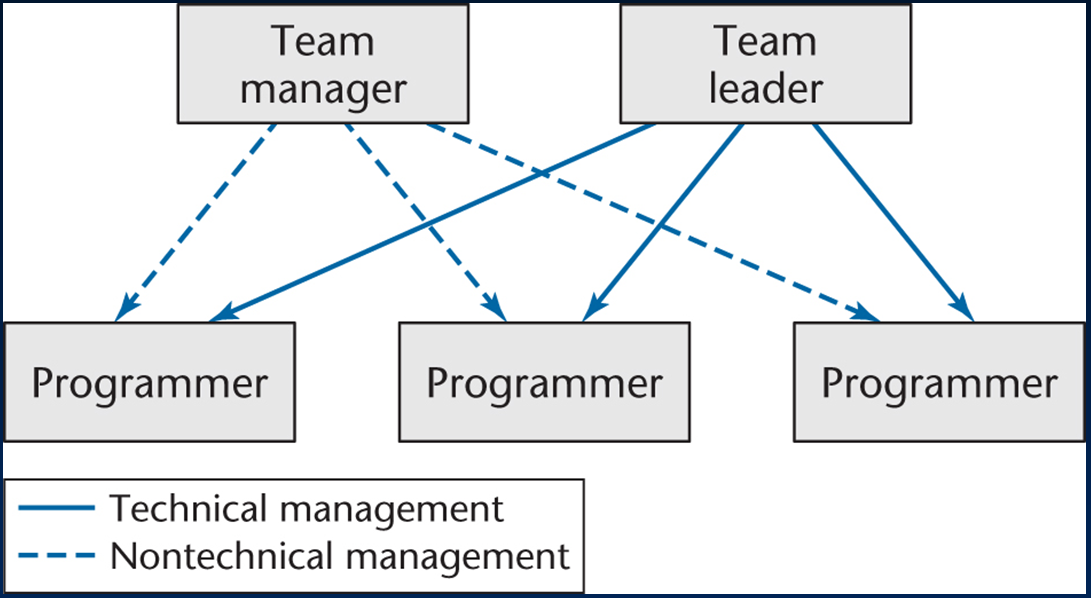
**CS 3402**

**Yuchou Chang**

## Group Structure

Our group started with the notion of completing our project under the democratic group structure, the structure later evolved into a hybrid team leader and team manager model to reduce the role of the chief programmer. Carlos Rios was appointed Team Leader as he had the technical knowledge to manage the rest of the programmers and the collaborative insight to create a final working code. Ivan Sosa took on the role of team manager, his goal was to handle the non-technical responsibilities that the group faced such as when to meet, organizing documentation, and overall group wellness. Below diagram details the management paths taken by the team manager and the team leader.

#### Hybrid Model

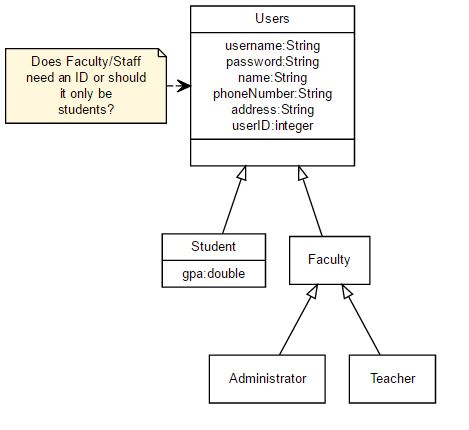


* Team Leader Carlos Rios
* Team Manager Ivan Sosa
* Programmer Satish Subramanian
* Programmer Phillipe Hehn
* Programmer Derek Jones

## Diagrams

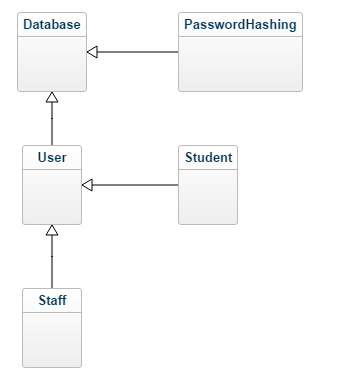
### Class Diagrams

#### Class Diagram – Version 1



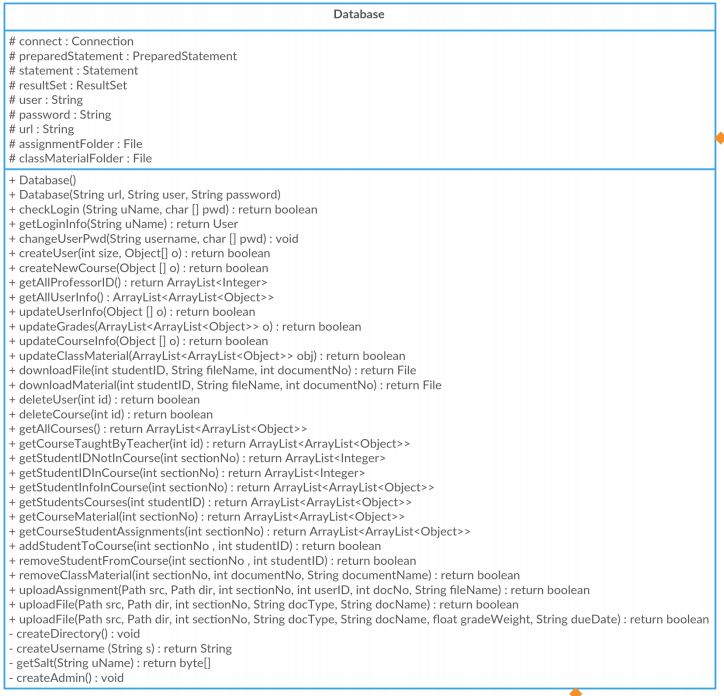
Above is an early class diagram made around the time of the group’s inception. Outlined is the users for the program and what keys needed to be created and tested for each type of user. Conceptualizing what was actually important features of a user class led us to later tweak and change our initial idea of the database. This users class concept will later be implemented into the table structure of multiple classes in our database.

#### Class Diagram – Version 2

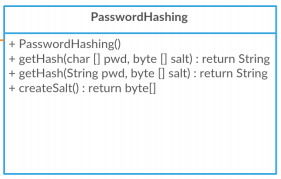


The above version of the class diagram demonstrates the hierarchy of the classes in the database. The main difference is that the administrator and teacher classes have been consolidated into the staff class. Password hashing, a one-way encryption model that maps data of an arbitrary size to a data of a fixed size, was talked about and implemented as a class in this version. The benefit of adding a password hashing system would be to reflect the real-world dangers of storing raw passwords of any form in a database unprotected.

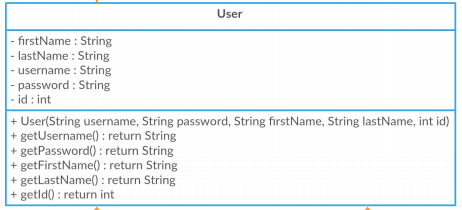
**Database**



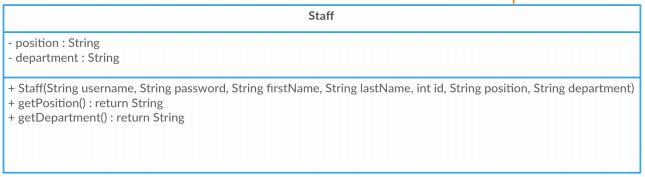
**PasswordHashing**



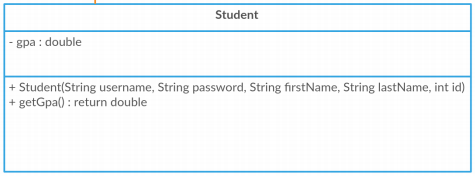
**User**



**Staff**



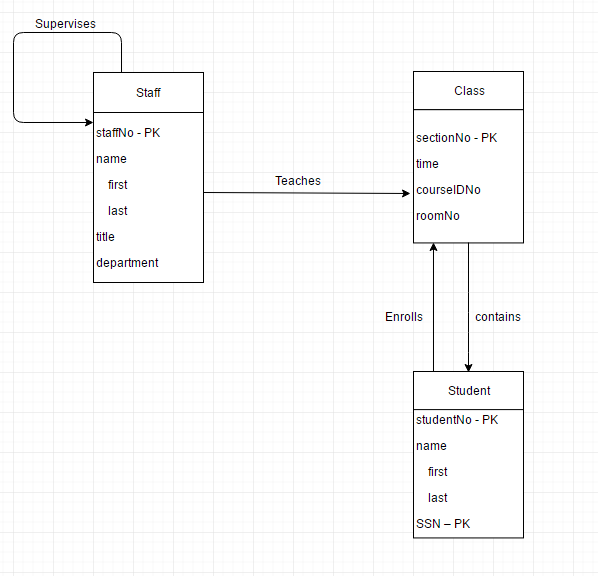
**Student**



Above tables show the classes of the Java end of the project. The first large box under the class title details the variables of the classes, while the second large box shows the functions that can be invoked from each of these classes. The database class of the java program establishes a connection to the SQL database to pull the data from it and establish the GUI.

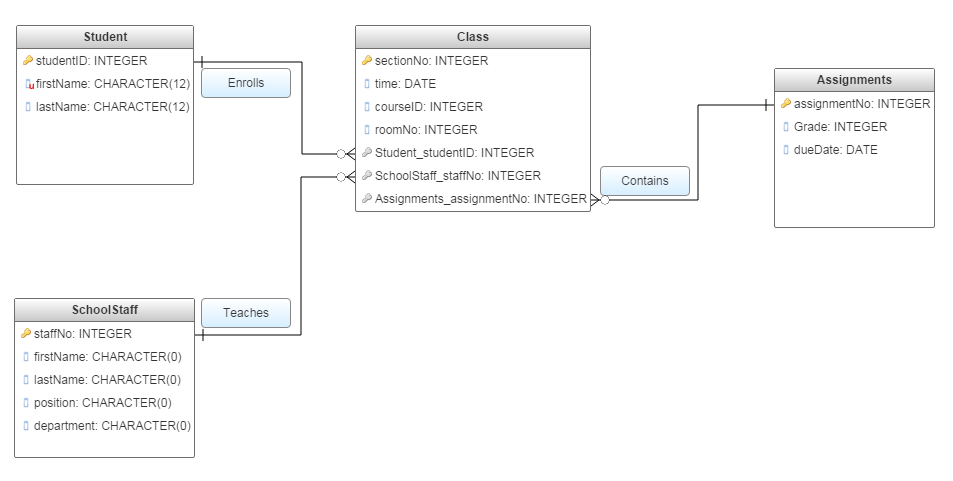
### Entity – Relationship Diagrams

#### ER Diagram – Version 1



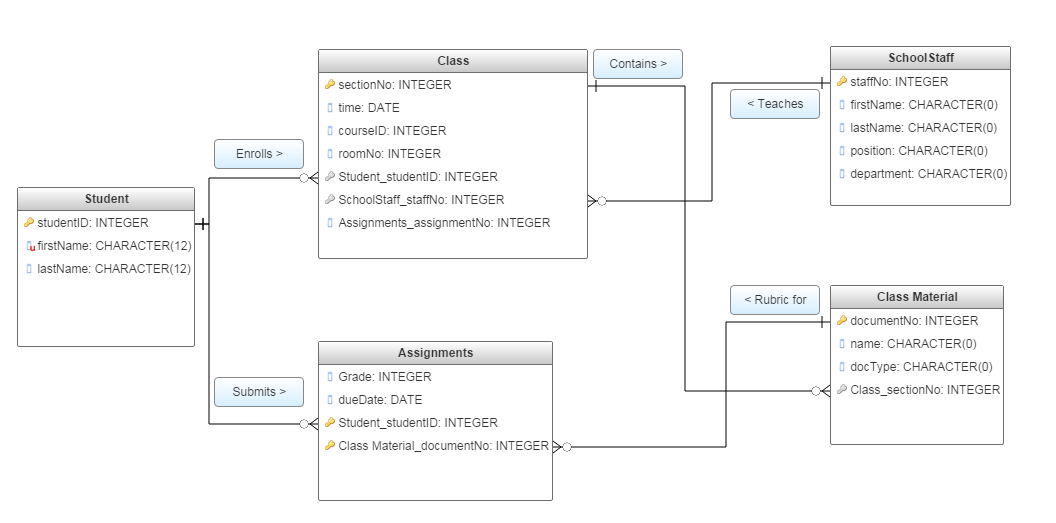
Our first version of the ER Diagram had a basic outline for how our entities would interact with one another. This initial ER led us to the creation of many more entities to handle the realized complexity of the project. It can be seen that this ER was created before the conceptualization of our GUI or the implementation of an assignment turn in system.

#### ER Diagram – Version 2



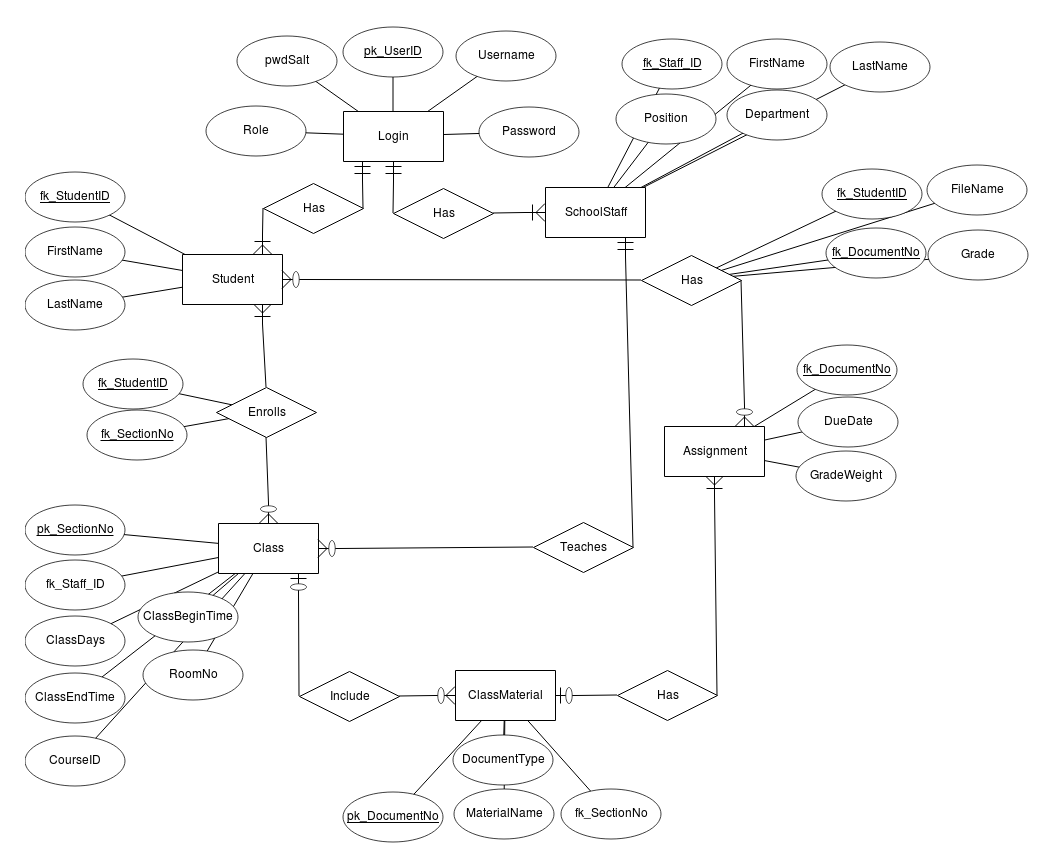
The second version of the ER had clearly defined relationship between entities comparatively. The data types that make up the columns of the tables were updated to better suit the project. This diagram also lacked a way to relate assignments back to the students who submitted them. In an effort to recreate one of the main functionalities of file upload, we decided to introduce another entity to our next version of the ER that would handle this.

#### ER Diagram – Version 3



The third iteration of the diagram had a way to handle a unique assignment turn-in for the class materials. After implementing the Class Material table to the database we had to add unforeseen post implementation functionalities. The schema for this class had to be modified to be more functional and the documentation for the ER had to be revised

#### ER Diagram – Version 4

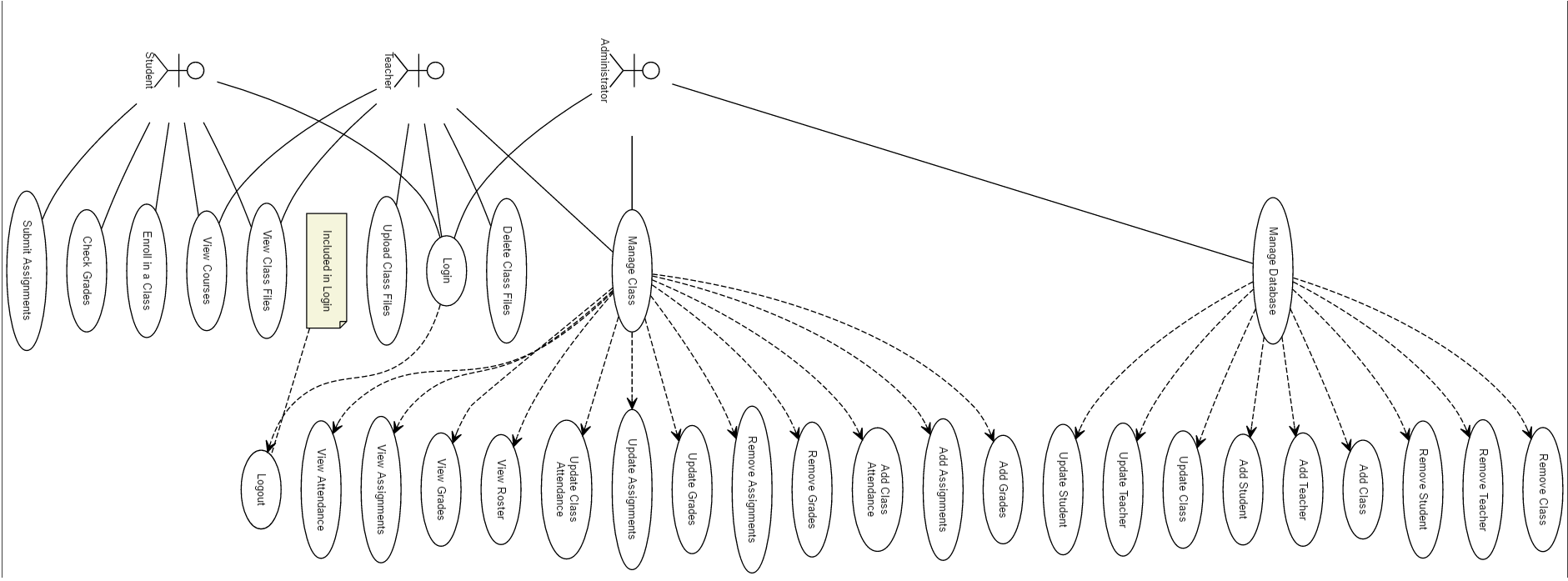


The final iteration of the ER diagram included a way to handle a login and a more functional class material table. We ended with 6 entities: Login, Student, SchoolStaff, Assignment, Class and ClassMaterial. This diagram represents the current schema of the completed project.

### Use Case Diagram

The use case diagram contains all actions that could be taken by users. The 3 main users were the Admin, Student, and Teacher.

Teacher needed to be able to add, modify, and remove assignments and grades. They also need to be able to view the class roster, as well as upload files and delete files.



Student uses cases were to view reports, course assignments, grades, and uploaded files. Students also needed to sign up for classes and submit assignments.

Later on we would merge the duties of teachers and admin and as a result their use cases would be a union of these functionalities.

## Conclusion

Our group easily progressed through the requirements portion of the development process, however we encountered issues during the analysis portion. Our project grew and evolved into much more than we initially planned for it to be. Every major functionality decision led to our diagrams and documents needing updates. For these functionalities planned for in the requirements stage to be realized the programmers needed to come together as a group to efficiently present a solution. Even though we spent more time than anticipated on analysis, we progressed through implementation quickly as the details were ironed out in the analysis phase. We as a team learned the importance of time management and collaboration in the face of deadlines, requirement dilemmas and organization. We are happy to be able to produce a documented final product ready for client testing.