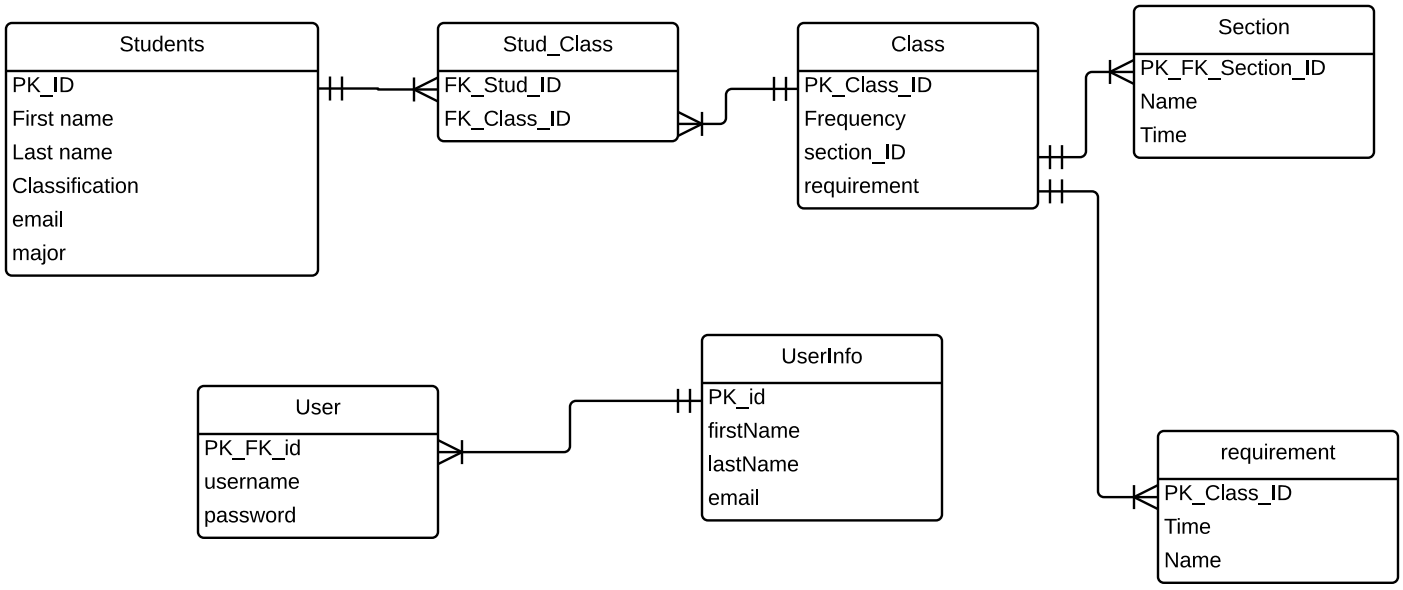
Class Schedule

Entity Relationship Diagram



Software Engineering Design Beta

Data input: Banner, desired class data and time from user.

Output: A list of students with class conflicts at the desired time, the conflicting classes, and possibly a hierarchy of class priorities/alternative times.

Compatibility with other systems: Not at the moment, this design deals with primarily Banner as it will be our source for student and class data for the foreseeable future. Further design for new systems may be considered at a later stage.

Likely changes: Since we are limiting ourselves to only getting input from Banner, the only change would likely be the format in which Banner presents data.

Product family: Web-based. We are going with a depth-first approach, because this program should only be accessible, at least at first, by advisors, degree plan specialists, and department chairs. These people can easily access a web-based system, and other families, such as mobile, should not be required.

Modularity: To prepare for the possibility of multiple data input possibilities, the Adapter pattern can be used to adapt incoming data to an expected format.

Mechanism vs Policy: The policies by which the mechanism we are designing adhere to are defined by the priorities in which different classifications of students have over each other. A student with a higher classification will have a higher priority in terms of class change compared to a student with a lower classification. The frequency of which a class is offered is also a concern with regards to policy. One that is infrequently offered has a higher priority over one that is commonly offered.

Model: Client-Server. This model was chosen as all data comes from Banner, making it an ideal candidate for our server. The Client is our program, which pulls data from Banner, takes ideal class time input from the user, then outputs a list of students that have conflicts.

Pattern Used: Model-View-Controller (MVC). Model is how our data is represented within our system (MySQL Database Backend), the View is our web front-end, and our controller is our business logic that will help determine the list of conflicts, and generate a database query to access that list. The controller also determines valis class times by querying the database.

Database Design:

1. **Table of students.**
2. **List of classes.**
3. **Transaction table linking students and classes. (One-To-Many)**

Student Table:

- **ID**
- **FName**
- **LName**
- **Classification**
- **Email**

Class Table:

- **CRN**
- **Name**
- **Time**
- **Frequency (Integer value listing priority)**

Transaction Table

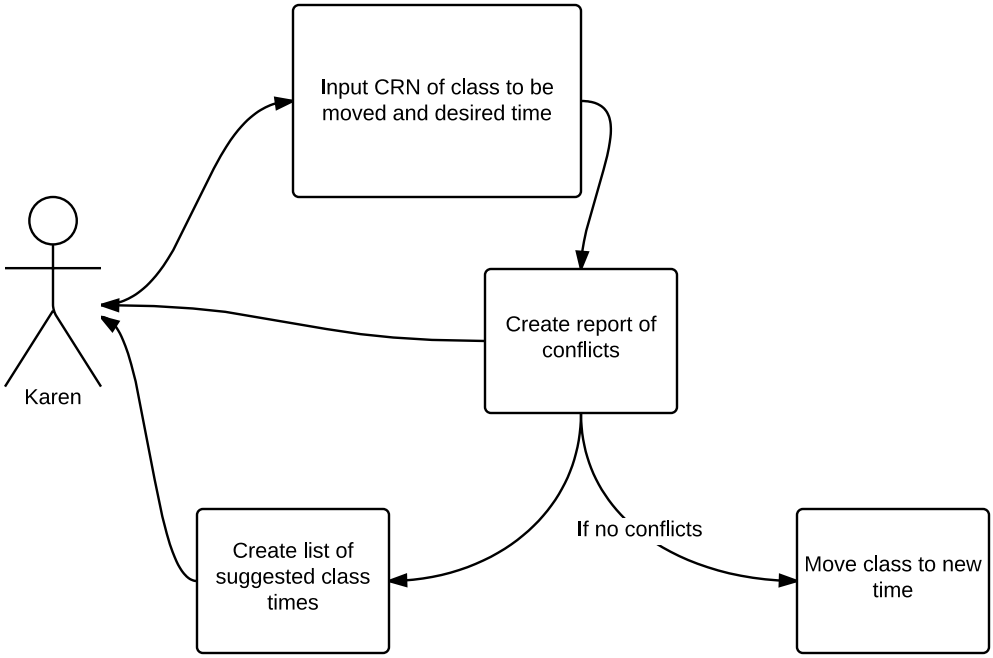
- **Student ID**
- **Class ID**

Frontend:

1. **Login page (Powered by MySQL Database of Users and PHP scripted).**
2. **Once logged in, User enters CRN and chooses from a dropdown list of available class times.**

Use Cases:

1. **Enter desired class and time and get list of conflicts.**
2. **Move classes to desired time.**
3. **Provide calendar to suggest a better time for the class**


Input CRN of class to be moved and desired time
Karen
Create report of conflicts
Create list of suggested class times
If no conflicts
Move class to new time