**Scheduler Algorithm Brainstorming**

2 Conditions:

-Prereqs

-Time slots

Take batch of 5 classes at a time, following classic course sequence.

If one class is impossible to put in schedule, check next 5 courses.

For basic Schedule:

First working schedule generated.

For least amount of breaks (and inverse for most):

Do a sum method of all time slots between courses that are not allocated to a class. Schedule with smallest break time during week is returned to user (or inverse if big breaks are wanted).

For early-bird schedule:

Do an average method, schedule with lowest average starting time and lowest average finishing time of every possible schedule returned to client.

For night-owl Schedule:

Same as early-bird but take latest start and finish time.

**\*Early-bird and night-owl schedules will by definition have as short amount of breaks as possible.**

How to keep track of Prereqs:

**To add to DB:**

1. Field with required courses for each course, those with no prereqs should have n/a or something similar but constant.
2. Field with completed or not for the course, simple Boolean will be good.

Possibilities to check if prereqs are satisfied:

1. Check through every completed course, if it is marked as a prereq in another course’s prereq field, erase it from that field so like that the scheduler only takes courses with n/a for prereqs (**so in this case n/a will stand for no prereqs or all prereqs completed**)
2. When the scheduler runs, every time it goes to take a course, it looks at the prereq list, if any prereqs are present, it finds them in the db and checks if the “completed” field is true.

Possibilities to avoid time conflicts:

1. Make sure Start Time and Finish Time are two different fields in the DB. After the first course is added, each course added checks:

Is Start time bigger or smaller than finish time of course? (Do for each course already added) If smaller, than is finish time of course being added smaller than start time of already present course? If start time is bigger than finish time, then also good.

Every time a course is added, create an array of Booleans the size of the number of already created courses. Every time the added course satisfies the condition for an already present course, the array at the index of the already present course is changed to true. If at the end, the array is true at every index, than the course can be added safely.

1. Keep a map with:

* Keys: integers that refer to every block of 15 minutes in the school day, for example, 0 refers to 8:15 to 8:30, 1 refers to 8:30 to 8:45, etc.
* Values: Boolean, if the block is already taken for a course, it is marked as true, if it is available, it is false.

In this method, courses will not have start and finish times but a list of every 15 minute block it requires. When a course is added, the blocks in the map are marked to true. Then when we want to add a course, we must require it to be placed in blocks with “false” value.