**graduation logic**

stuff we get from query

* username
* desired\_gpa
* graduating
* courses 1-7

input list = list(courses 1-7)

(courses only added if they’re not None)

1. **sanity check to see if what they’ve already taken + what they want to take satisfies req’s**

select crn from taken where taken.username = 'username' union select crn from courses where cname in [input\_list];

**pass this result thru graduation function**

* if they can graduate with this schedule, then maybe pass it on to improve it
* caveat: can’t mess with specialized classes in this case

2. **create view spec\_courses for courses in user’s specialization**

(note: specialization didn’t have proper column names, so I’m inferring what these would be)

SELECT crn, cname

FROM user

JOIN specializations ON specialization

WHERE user.username = ‘username’

3. **find which courses in input list are in user’s specialization, create view include\_grad**

SELECT crn

FROM spec\_courses

WHERE spec\_courses.course\_name in ‘input\_list’

4. **find which courses in their specialization *aren’t* in input list as view non\_include**

SELECT course#

FROM spec\_courses

WHERE spec\_courses.course\_name NOT IN ‘input\_list’

**5. Validate graduation requirements**

**i)** query to get comprehensive list of what they’ve taken + courses from their

input\_list as one giant list

* this list is “input\_specs”
* should be possible from views we just created

**ii) we get a list of all possible k-sized schedules of only specialization classes**

* k here is length of our original input list (e.g., 4) minus the # of specialization courses from input list
* E.g., if user wanted to take 7 courses overall, and 3 of the courses they input were in their specialization, then k = 4
* This is because k = the number of unnecessary courses here, which means we can replace them

**iii) we find all of these that don’t have time clashes**

* if don’t satisfy time-clash query check, continue through loop

**iv) we pass each of these through graduation requirement function**

* add each of these that passes to running list of schedules which satisfy grad requirements

**v) if any satisfactory schedules found:**

* find which of these has the best overall gpa

**vi) if none found:**

* get list of ALL classes from user’s specialization being offered, INCLUDING ones from their inputted schedule
* repeat the process as above to generate all possible schedules
* if still no satisfactory schedules are found, then the user cannot graduate with however many classes they wanted to take

In short, we want to see if we can satisfy their graduation requirements with all the classes they wanted to take that were in their specialization. We do this by trying every combination of these classes and all other classes being offered in their specialization. If none of these combos (along with what they’ve already taken) allow them to graduate, then we just try all possible combos of courses being offered in their specialization this semester (including the ones they wanted to take). If none of these satisfy their requirements, they can’t graduate with that amount of classes.