Design Notes - Backend

Tuesday, April 18, 2023 6:13 PM

Class settings

- group size
- extra skills to add
  - ↳ skill name
  - ↳ 'required' number per group
- prevent gender isolation checkbox
- prevent race isolation checkbox

Student Form:

- schedule availability: how is this stored?
- for each skill:
  - need improvement
  - neutral
  - confident

  } could also just be stored in backend as a number

- Personality Section:
  - store two personalities
- gender - M/F/NB
- name - string

Scoring:

- schedule availability: # of hours everyone can meet (normalize somehow)
- skill section: % of skills which the team is confident in.
- personality section:
  Sum for each personality option
  - max(0, # of people in the group with that score - 1)

Classes/Data-Types

- **CourseStudent**
  - → firstName
  - → lastName
  - → studentID
  - → gender (str)
  - → ethnicity (str)
  - → availability (?)
  - → skillRankings
    - ↳ HashMap
      name → ranking
  - → personalValues
    - ↳ array of strings

Course

- → courseID → areGroupsPublished
- → students
- → groups - start empty
- → skills - start w/ defaults
- addSkill(SkillRequirement)
- makeGroups(int size, bool prevGIso, prevRIso)
- moveStudent(studentID, newGroupID)

SkillRequirement

- name
- numRequired

Group:

- → groupID
- → students
- removeStudent(Student s)
- addStudent(Student s)
- score

Initialize to undefined (get from form)

Idea:

store as lists of time blocks

TimeBlock

- → day (int, 0-6)
- → startHour
- → endHour
- overlap(timeblocktb): TimeBlock | undefined

Schedule

→ TimeBlock[ ]

- merge(Schedule s): Schedule
- totalHours: number

Ideas for group generation algorithm

- Make like 50-100 sets of groups which match valid criteria
  → use the one with the highest score
- start from scratch, add each student to the group which would benefit the most (and is valid)
- ~~x~~ randomly generate 1 set of valid groups, then perform random swaps
  → If the swap score stays the same or goes down N times in a row, then a stopping point has been reached