Full stack application idea: Student management

* An associate can have many assignments
* Two roles: trainer and associate
* users table w/ two roles trainer and associate
* assignment table (one user can have many assignments)
* comments table (one assignment can have many comments)

User Stories

* Trainer can assign a due date to assignments
* Associates can submit assignments in the form of a file
  + Submission must be before due date/time, otherwise it will be rejected
* Associates can view all of their own assignments
* Trainers can view ALL associates’ assignments
* Any trainers can assign a grade to the submitted assignment
  + grade will be out of 100
  + can also edit grade
* Trainers can add a new associate
* Associates can see their grade average
* Trainer can remove an associate
* Trainers can submit comments on an assignment
* Users can log in/log out

P1

* 2 tables: Users and reimbursements
  + User table has 2 roles: employee and finance\_manager
* One user can have many reimbursements (author (regular employee) and approver (finance manager) <- 2 FK pointing to the users table)

User Stories

* Users can log in/log out
  + Endpoints:
    - POST /login
      * { “username”: “user123”, “password”: “password12345” }
    - POST /logout
      * No request body
* Employee Users can submit reimbursements w/ a receipt file
  + POST /reimbursements
    - On the backend, will figure out which user is logged in “currentuser” attribute and add the reimbursement with the author being the logged in user
    - Request body
      * { … } JSON?
      * form-data?
* Employee Users can view their own reimbursements (but not other peoples’ reimbursements)
  + GET /reimbursements
    - On the backend, will check if the user that is logged in is a regular employee, and if they are, retrieve only THEIR OWN reimbursements
* Finance Managers can view ALL reimbursements (and filter by status, PENDING, APPROVED, DENIED)
  + GET /reimbursements
    - On the backend, will check if the user that is logged in is a finance manager, and if they are, retrieve ALL USERS’ reimbursements
* Finance Managers can change PENDING reimbursements to DENIED or APPROVED
  + PATCH /reimbursements/{reimbursementId}?action=approve
  + PATCH /reimbursements/{reimbursementId}?action=deny
    - Both of these are the same endpoint, we just need to check what the value of the action query parameter is