CS 168 FA24 Lecture Attendance Checker

Built by peyrin@berkeley.edu in Sep 2024 to track lecture attendance.

If we gave out one code word during lecture, students could trivially send the code to their friends, and now their friends can claim attendance even though they didn't show up.

To mitigate this, we distribute distinct one-use-only codes to students at the end of lecture.

This system should also work if you give out a single word. (For example, if a lecture runs late, you might just give a single word instead of distributing codes, so that students can clear out of the lecture hall quicker.)

Form

- The CS 168 FA24 form.
- Optional: Regex the Student ID to prevent typos.
    - Berkeley student IDs are always 10-digit numbers starting with 303 or 304, or less commonly, 8-digit numbers starting with 1 or 2.
- Optional: Instead of displaying all the lectures in a huge list, add options for upcoming lectures and remove options for very old lectures throughout the semester. Requires semi-regular upkeep of the form.
    - Also, students need a way to submit old codes if they forgot to do it at the time - maybe reopen at the end of the semester with all lectures listed again?
- Highly recommended: Regex the attendance code to prevent typos.
    - We use 11 lowercase letters. You can use something else if you want.

Generating Codes

- I used a Python script to do this, but I threw it away. Oops. Shouldn't be too hard to write another one though.
- However you end up making them, separate each code by newline, and make one separate file of codes per lecture.

Spreadsheet

- The CS 168 FA24 logging sheet.
- For each lecture, make a new sheet.
    - Title: That lecture's number (e.g. 6, 7, 13, 27).

    - One column in that sheet should have all the attendance codes. Preferably the same column in all sheets. I used Column B.
- Make a sheet called "Form."
    - Link your attendance form to this sheet.
- Make a sheet called "Validating."
    - This sheet has all the form submissions (one per row), along with additional columns to validate their code. A student appears once per lecture on this sheet.
    - In A1, write a formula to pull all data from the form submissions.
        - Example: `=QUERY(Form!$A$1:$E)`
        - We need this because later columns have formulas, and if you try putting formulas directly on the Form sheet, they would get clobbered out by every new form submission (which rewrites the entire row).
    - In F, write a formula to extract the lecture number from the form submission.
        - Example: `=MID(Validating!D2, 9, 1)`
    - In G, write a formula to check if the code is valid.
        - Example: `=NOT(ISNA(VLOOKUP(Validating!E2, INDIRECT(F2 & "!$B$1:$B"), 1, FALSE)))`
        - The INDIRECT lets us evaluate the lecture number as an expression and go to that corresponding sheet.
- Make a sheet called "Roster."
    - This sheet has a list of all students (one per row), and tallies their attendance.
    - In D, write a formula to sum up all subsequent columns.
        - Example: `=SUM(E2:2)`
    - In E, F, etc., make one column per lecture.
        - Title of column (E1, F1, etc.) should be the lecture number, e.g. 6, 7, 8, 9, etc.
        - In the column, write a formula to check if the student attended that lecture.
        - Example: `=COUNTIFS(Validating!$B$2:$B,C2,Validating!$F$2:$F,$E$1,Validating!$G$2:$G,TRUE)`
        - This checks for a row matching: Student email (first part), lecture number (second part), a valid code (third part).
    - Important: This sheet needs to be the first (left-most) sheet on the spreadsheet, or else the autograder breaks.

Printing Codes

- One Google Doc per lecture.
- [CS 168 FA24 Lecture 6 example.]
- Recommended: Use monospace font to avoid ambiguous letters in the code.
- Recommended: Decrease top/bottom page margins (File → Page Setup) to print more codes per page.

- TODO: At the current font size, all odd-numbered pages align (so you can cut them at once), and same with all even-numbered pages. There is probably a way to get all the pages to align, but whatever.
- Yes, you have to cut them into slips by hand. It shouldn't be too bad as the semester wears on and fewer people show up to lecture.

Autograder (Attendance Checker)

- This uses https://github.com/cs161-staff/ag-frontend as a base.
- Clone the ag-frontend repo.
- Follow these steps to get Google Sheets credentials: https://docs.gspread.org/en/latest/oauth2.html
    - Do the "Enable API Access for a Project" steps.
    - Do the "For Bots: Using Service Account" steps 1–7 (to get the JSON file).
    - Put this file in `ag-frontend/keys/google-service-accounts/credentials.json`.
    - Share the spreadsheet with the client email.
- Replace `generate.py` with the code in `engine/examples/attendance.py`.
- In the new `generate.py`, replace `spreadsheet_url` with your URL.
- In the new `generate.py`, edit `rows` with your lectures.
- Run `make autograder.zip` to create a ZIP file.
- Make a new Gradescope submission with this autograder ZIP.