COL732: Project

We will do a collaborative project where the whole class works toward one product. The product we are going to create is a cloud service for teaching and evaluations using vmm-reference. Using this service, instructors can upload assignments and a grading rubric by the means of automated tests, the students can implement their solutions and submit them, and the TAs can evaluate the submissions. Example features:

- Isolation: Students can do anything inside the VM without hurting other students.
- Snapshot/fork: all the installed software, files, memory state, etc can be restored from a snapshot. This is useful for an instructor to set up a snapshot and let the students fork it.
- Logging events: To catch cheating, log interactions to the VM such as (1) timestamp of changes being made to files and (2) all the processes being launched. These interactions can be later evaluated using a script to catch spurious behaviour.
- Consolidation: We would like to support many students on a single physical system. Can we let them share pages and files safely?
- Webserver exposing APIs for forking/starting/stopping.
- Frontend for actually interacting with the product.

The project has the following OKR (Objectives and Key Result):

The project will implement a *usable* teaching and evaluation hypervisor as measured by conducting a Rust basics lab assignment for all the students in the course.

Execution

To do the project, we will recruit leads. These leads will get the experience of "tech leads" in a software company. Becoming a lead is voluntary. You can nominate yourself by filling out this sheet.

Leads will work together to create a cohesive plan for creating this cloud service. They will further break down this problem into smaller problems. Each lead picks a problem they would like to be responsible for. One way to break down the product is through the features above. Another is to own user flows. One subproblem is the "instructor view": creating an assignment and submitting a rubric. Another is the "TA view": managing resources and scheduling evaluations. Another is "students working on assignments". Another is "students submitting assignments". And so on. Leads are free to choose any breakdown as they see fit.

Leads will also define their OKRs such that it aligns with the overall OKR. OKRs, by their definition, have to be ambitious. Realistically, you should only be able to do 70% of the promise. Leads will also justify why the OKR is ambitious for the number of team members working on the OKR. For instance, front-end development is not ambitious for a 10-student team. Example OKRs:

- 50 students can simultaneously code on a single physical server.
- Creating an assignment and its rubric takes < 2 minutes.

The team leads will then present the cohesive plan to the class and attract other students to their projects.

Grading

Full class:

- Hitting overall OKR gets +25% to everyone. Overall OKR is hit if we can *smoothly* run the lab assignment.

Team:

- Hitting team OKR gets +25% to everyone on the team. Reflection about how well the team OKR is hit is mentioned in the final project report.

Individual:

- Every team submits a report. There is a section where each individual mentions the work they have done.
- The team lead distributes 100 tokens to everyone on the team including themselves with justifications for the split. Other members can debate with the team lead, but the team lead's decision is considered final. In rare cases, where conflicts continue, it can be brought to the attention of the TAs and instructor. Make sure that you keep the provenance of all your work: write designs in Google Docs and write code in git. If the token split was found justified by TAs, a 10% penalty is given to the contender.
  - Token counts will be used to give the remaining 50% of the grades.
  - Other metrics will also be considered here such as the degree of contribution and clarity of presentation.

Special bonus for team leads:

- Get +20% for all the extra hard work if the team hits 0.8*results in OKR. No questions asked.

Deliverables– Project proposal

The proposal should contain the names and entry numbers of all the students in the project group. The proposal document shall be less than a page long. Here, you can describe your OKRs and why you think they are challenging to achieve.

You can declare your group here. You can start working on your project right away after submitting your proposals. In most cases, the proposals will be accepted as is. If there are some comments, they will be communicated to you and you can course-correct your project execution.

Deliverables– Final

Towards the end of the semester, each group will give a project presentation to describe what they set out to do / what they ended up doing / major roadblocks / and final learnings. Each project member should talk about the work they've done. Some tips on communication.

The slides along with a brief project report should be combined together as a PDF and uploaded to Moodle. The combined PDF should contain the names and entry numbers of all the students in the project group and a link to your code (such as GitHub PRs). It should also report the work done individually by each group member.

Timeline

15 Sep: Release project
22 Sep*: Project brainstorming
29 Sep: Finalization of team leads
6 Oct*: Project proposal presentations by team leads
9 Oct: Project proposal submission. Contains team members and OKRs.
30 Oct: Release Rust basics lab to all students on the developed product.
4 Nov: Close Rust basics lab submissions.
7 Nov*: Project final presentation (1)
10 Nov*: Project final presentation (2)
11 Nov: Project final report submission

*: These activities happen during lecture hours

Late policy

There will be no make-up presentations. No project material will be accepted after 11 Nov.