20875: Software Engineering

Project report guidelines

Fall 2025

What to upload

For the project, upload all of the following to Blackboard:

1. Your code:
   This can be an archive (e.g. tar or zip), or a link (e.g. to a public repository like GitHub or GitLab).
2. A short report (see details below):
   It can be written directly on Blackboard, or an attached document (e.g. text, pdf or MarkDown), or a link to such document.
3. Relevant material if any:
   Input data, example output, screenshots, videos, ... (uploaded directly, or as a link)

Short report

Regarding the short report, it should be about a page long (there is no hard minimum or maximum, but a page should be plenty enough). As a general rule, the purpose of the short report **is not** to

- explain how to use or install the project
- explain what specific functions / classes / modules do
- describe the individual steps of the algorithms / methods in use.

All of the above, if required, would be a separate documentation. **Instead, the short report should present:**

- A paragraph explaining what your project does, in which language it is implemented, and what context it is intended to be used in (e.g. a system library, a device driver, a server-side web application, a server administration tool, a mobile app, a desktop application, an inference engine, ...)
- If relevant, the general components of your project (e.g. a GUI module + back-end logic, or preprocessing pipeline + main algorithm + various performance-improving heuristics, ...)
- Any hurdle or interesting problems you encountered. Even seemingly trivial things can be interesting: For example, installing a library your project depends on may be complex, and you may have found a systematic way to make it work. Or, you may have managed to get data from a source that did not intend to provide machine-readable data.
- Any ideas you came up with. Those ideas may be fully implemented in your project, or just as a proof-of-concept.

About LLMs. The point of the report is to attract our attention to the interesting aspects of your project. If a report is seemingly written by an LLM, that makes it look low-effort, and reflects correspondingly on your project.

Example

Note: the text below is just an imaginary example, and its structure may not be a good fit for your specific project. You are free (and encouraged) to structure your report in a different manner.

We made a routing application for dog walkers. It is an iOS app written in the Swift language.

The main components are (i) the user interface, (ii) a Google Maps API query module, (iii) a routing engine.

The UI has three panes, one for the map area configuration and general user settings, one for the dog data entry, and one main pane showing the map with the current position and the advised route. For each dog, we can encode the pickup and drop-off coordinates, the pickup and drop-off time windows, and a list of fellow dog exclusions (some dogs just won't get along).

We encountered issues with the Google Maps API: For n dogs, we need to query distances for a full n^2 distance matrix. As such, we easily hit the query rate limits for the free tier of the Google Maps API. For now, our solution is to ask the user to request a route computation at least two hours in advance, so that we can spread our queries over time.

The route computation is not trivial. The problem is similar to (but distinct from) the "capacitated vehicle routing problem with time windows", and it turns out to be NP-hard. We model it as a mixed-integer programming problem and solve it using an off-the-shelf branch-and-bound solver. However, out of the box, the solver was too slow for practical purposes (multiple hours for our randomly-generated 60-dog example instance). To overcome this, we opted for a standard column-generation formulation, as generally used in the vehicle routing literature. The complexity here came from our fellow-dog-exclusion constraint, but we managed to modify an existing pricing algorithm to take that into account.

As currently implemented, our application lacks many of the features we initially planned for (bark recognition, trash bin avoidance, ...), and the user needs a jailbroken iPhone for sideloading. Also, it tends to drain the phone battery in a couple of hours, which we plan to solve in a future release. However, it works mostly well enough for the smaller dog walks.