Software Engineering
- XP Phases -

P. Baumann, Jacobs University

After the design phase is finished we now start with implementation. This is done using "Extreme Programming" (XP) and "Pair Programming" (PP), organized in bi-weekly "sprints". The TAs will randomly assign teams and their code bases, making sure both change with every sprint.

The following sprints are foreseen:

- Sprint 1 – deadline: 2022-mar-18
- Sprint 2 – deadline: 2022-apr-01
- Sprint 3 – deadline: 2022-apr-22
- Sprint 4 – deadline: 2022-may-06

Your task is to advance the code base you have (which initially is empty, of course). Proceed implementing along the design chosen, under the following rules:

- Any progress counts, such as: more functionality (backend, GUI), more tests, more documentation, as well as any improvements thereof – in particular: improved source code wherever adequate.
- The group's sprint results will be fetched from the repository shortly after the deadline, as the single source of truth (no email submission etc.).
- Provide a README document listing your advance; **we will use your README to assess your progress** – we might find undocumented progress, but no guarantee (and no complaints will be accepted in this respect).
- Every sprint outcome will receive percentage. The final grade is the average of all components.
- Any questions: during the project sessions, via list (no responses to questions in private!), or in class.

Likely the design is not perfect. Should you find an error or omission then ring an alarm bell, meaning:

- Point out the problem and suggest a change to the specification, consisting of a new document version where changed parts are marked clearly.
- Follow that changed spec in your current sprint.
- Make sure the new document version with its editable sources are in the repository.
- If the TAs approve it during evaluation then your version becomes the new design binding for all starting with the next phase. Note that several change proposals might need to be merged, so make sure you provide the editable sources of your document.
- An approved change request is an achievement and, as such, will earn points.

- If in doubt ask the TAs ASAP during the sprint; they cannot provide the solution but potentially clarify issues on how to produce and document changes.

So…let's roll and get our hands dirty! ☺

Evaluation Criteria

(as outlined in class, with explanations)

- Code runs?
    - checked by invoking the deployed (!) website + inspecting source code, at our discretion
- Amount of progress overall, based on README file
    - Features implemented / extended, GUI, code quality like formatting, documentation, code quality (see below)
    - Just to repeat: evaluation can focus on README documented progress only, anything not documented may or may not be considered – make sure all progress is documented!
- Code quality, with criteria such as (!) meaningful class structuring and interfaces, exception handling, correct output formats, comments, proper formatting, meaningful variable & function naming (avoid 1…2-char vars!), ...
    - Avoidance of global variables unless absolutely necessary and extensively justified
- Amount & quality of test cases
    - Test code is code, therefore subject to the same quality criteria!
    - Have as many test cases as ever possible, the properties that can be tested in a program are virtually countless
- Amount and quality of documentation
    - Inline documentation (checked automatically), external (PDF) documentation
- Plagiarism check
    - checked automatically
    - Codebases with a significant amount of plagiarism will immediately receive the lowest possible score.

In summary:
sprint result = features implemented + tests implemented + GUI + documentation + visual code quality

Some practical guidance

- The README is the entry point for evaluators, feed it comprehensively! Just to restate.
- You cannot overestimate the value of a large set of test cases! Don't forget to document the test code – it is code, so all quality aspects apply (just to restate that)
- Software requirements: make sure to list any particular requirement necessary to run your code; base requirement in any case: server-side on CLAMV, client-side any browser we might use for evaluation.

- using a non-recommended framework or library does not pay off - others will have to live with it, and you will not see your code again anyway, so you will have to immerse into a completely different architecture rather a just somewhat different implementation
- design change suggestions: if meaningful earns points even if not accepted into the new design version
- access protection: if you decide to protect access to your service make VERY sure that for evaluation we can lift that based on your documentation (user, passwd) without any search & trial – we won't do that.
- Make sure that your code is deployable anywhere! Best test deployment on "the other" partner's public_html/