[CS304] Team Project - Sprint 2 (20 points)

> Presentation (no rearrangement):
>
> - Week 15 lab session, +0.5 bonus
> - Week 16 lab session
>
> Report submission: Before presentation.

Part I. Team Report (9 points)

You are required to write a team report, which includes the following information.

1. Metrics (1 point)

We could use metrics to describe the complexity of your project. Please compute and report the following metrics (you could use whatever tools or libraries for computing the metrics):

- Lines of Code
- Number of source files
- Cyclomatic complexity
- Number of dependencies

2. Documentation (2 points)

A well-documented software system is easy to use and maintain. Please demonstrate that your team project has the following two types of documentation:

- Documentation for end users: this could be a `readme` or a `wikipage`, which provides essential information or steps of using the software. End users could use this type of documentation as instructions or user manuals to use your software.
- Documentation for developers: this could be an API documentation that is intended to aid developers. By reading this type of documentation, developers, collaborators, or potential future contributors could understand the design, purpose, and implementation of important code entities (e.g., classes and methods), enabling them to update or extend your software with ease.

It's preferred that you provide URL links to the documentation in the team report (e.g., if the documentation is in `.md` or `.html` format and version controlled, you could provide the GitHub link to the resource).

You could also provide snapshots. Yet, if the documentation is too long or scatters around your project, DON'T put all of them in the team report. Instead, you could provide a few snapshots and a textual description describing the documentation.

3. Tests (2 points)

Your project should have **automated** tests for quality assurance. Please describe:

- The technology/tools/frameworks/approaches that you used for automatically testing the project.
- Source code or related artifact for testing. Put URL links (preferred) or snapshots in the team report.

- How effective are your tests? Please provide a brief answer and explanation of this question (e.g., you may use a test coverage report to showcase the effectiveness of your tests).

If, for certain reasons, you can only test the project **manually**, please describe the difficulty of test automation, and explain the detailed steps and approaches used in manual testing.

4. Build (2 points)

Your project should have **automated** build process to compile source code, assemble dependencies, and produce executables. Please describe:

- The technology/tools/frameworks/approaches that you used for building the project.
- Tasks executed in a build. In addition to compilation and packaging, you may also consider adding linters, tests, test report generation, and documentation generation as tasks of a build.
- The final artifacts produced by a successful build.
- `Buildfile` (e.g., `pom.xml`) or related artifacts/scripts used for building (again, provide URL links or snapshots of the buildfile).

5. Deployment (2 points)

A software system needs to be properly deployed to be accessible to the public. Containerization is one of the modern techniques that makes deployment easy.

Please describe:

- The technology/tools/frameworks/approaches that you used for containerizing the project.
- The script or related artifacts (e.g., `Dockerfile`) used for containerization (provide URL links or snapshots).
- A proof of successful containerization (e.g., you could provide a snapshot of a successful docker build).

If, for certain reasons, you cannot containerize the system, please describe the reason why containerization is difficult for your project. You should also provide a detailed description of the deployment approach used for your project.

Part II. Collaborations And AI Usages (1 point)

Complete this survey to report your AI usages for documentation, tests, build, and deployment. One submission per team is enough.

Same as the previous sprint:

- You should use git properly for version control and collaborations.
- You should annotate AI usages in your code.

See documentation of sprint 1 for details. **Failures to do so will incur score penalties.**

Part III. Sprint Review (10 points)

In this final presentation, your team should demonstrate the entire running system, focusing on features and usability.

Imagine that this demo will be presented to end users, clients, and potential investors. Your team should demonstrate all notable features of the system, with the purpose of attracting users/investors to use/invest the software system.

Steps of the demo:

1. Build the application to produce the executable binary. If this process takes too long, you should record a video of the building process. Also, just in case that anything goes wrong in the build, you should have the executable ready before the demo.
2. Start you application by the executable or from the container.
3. You should introduce each feature in a clear, ordered fashion. We require that the team project should support at least 5 distinct, notable features. Hence, you probably want to introduce these features one by one (e.g., the first feature is ...., the second feature is ..., etc.).
4. You DO NOT need to explain implementation details, since the audiences should be end users/interested parties with no/little technical background.
5. Briefly demonstrate the team report. Slides are not required.
6. The demo should be within 8 minutes. We'll have 2 minutes for Q&A.

Basic requirements (up to 50% points will be deducted for failing these requirements):

- Each team will give a presentation **in the lab session of week 15 or 16**.
- Every team member needs to show up during the presentation.

Submissions

We will check your team repo for the code, scripts, other artifacts, and the commit history of your project.

You should also submit the team report as `final-report-teamID.md` to the team repo.