Project Guidelines

Last update: 2025-11-14

The following guidelines specify how the up to **30 project points** can be earned in the context of the group assignment.

Summary
Students work in small groups (approx. 4 students each) to implement a *data science themed* software project that should[1] belong to one of the three following categories:

(A) Interactive visualizations of available structured data (e.g.: fetch data from a given API, process and visualize it depending on user supplied parameters in a dashboard or Jupyter Notebook)

(B) Data quality tools: software that detects inconsistencies or errors in (raw) datasets, validates raw data against a specified schema or carries out other forms of validation and consistency checks (e.g.: check some specified local or remote database tables for duplicates or missing values)

(C) Applications that process raw data and transform it, making it available in a structured form (e.g.: crawl websites or parse PDFs, store the data in a structured form and make it accessible via an API)

During the project phase of the course (end of November to end of February), the groups have regular, mandatory update meetings with the lecturer (scheduled individually per group) to discuss their progress.

The projects should be made available (publicly or privately) on GitHub[2] and should be setup as best-practice open source projects; details and minimal requirements are described below.

General distribution of points

The **30 points** in total are split into three parts:

- Up to **5 points** can be earned for the code itself,
- Up to **10 points** are awarded for the version control history, the project infrastructure (documentation, CI/CD pipelines, …), and the way how students used code platform features (issue tracker, pull request reviews, …) to collaborate,
- Up to **15 points** for a short written devlog, summarizing how the group has approached implementing their project from start to finish (conceptualization, writing specifications, coding, code reviews, releases), including descriptions of which group members had which responsibilities.

While the projects are group assignments, students within a group are not necessarily graded with the same number of points.

Timeline and important deadlines

- The initial deadline for forming (preliminary) project groups is **Friday, November 28**. There is a self-registration tool in our Moodle course, groups can be formed from students independent of their exercise group. Everyone who wants to pass the course **needs to be registered for one of the available groups until the end of Friday!**
  - Talk to other students and/or use the coordination forum in our Moodle course to exchange ideas and get an idea for who you would like to work with.

[1] Do suggest your project ideas anyways, even if they don't fit cleanly into any of the three categories.

[2] Separate repositories on university infrastructure are available on demand. On GitHub, the repositories should be created within the KFU-DATB31UB-25W organization, this will happen as part of the first update meeting with the lecturer.

  - If you don't want to assemble a group on your own, register for the group named "Random Assignment". Students registered for this group are assigned randomly after the deadline.
- On **Monday, December 1** preliminary group assignments are published. In this step, students from existing groups might be reassigned (especially if students with a prior computer science background are distributed too unevenly); generally this is a rather extreme measure and pre-formed groups are rather extended, rather than broken up.
- Objections and change requests are possible until **Friday, December 5** – after our lecture, the groups are fixed.
  - Groups should then meet to (a) fix the communication channels they want to use internally and (b) discuss their project ideas.
- Every group is expected to have an initial project description / specification that has been discussed with and approved by the lecturer until **Friday, December 19**.
  - It is the responsibility of the group to arrange a meeting for discussing their project idea.
  - The usual time slot of the course (Friday afternoon) is generally reserved for such meetings; it is also possible (subject to availability) to schedule a meeting outside of that slot and/or on another day.
  - As soon as the project idea is approved, a Git repository is created in our course organization.
- After a group has practically started working on their project, at least one update meeting should be scheduled with the lecturer in January to discuss the progress of the group.

Once a group has finished their project and the corresponding devlog (**hard deadline: February 28, 2025**), one member should send a link to a released / tagged version of the project together with the devlog (as a PDF) to benjamin.hackl@uni-graz.at.

Within a week (if possible), we will then arrange a final update meeting for a short, informal presentation of your project. Afterwards, the project points are distributed and grading is completed.

Minimal project requirements

- You are free to implement the project in any "higher" programming language of your choice: Python, Rust, C++, C#, Java, JavaScript, … are all okay, but your project has to constitute the appropriate equivalent of a Python package: there should be a (standardized) file in which metadata (version, authors, dependencies, …) is specified (`pyproject.toml` in Python).
- Avoid committing any unnecessary files (automatically generated files, caches, secrets, …) to your repository. Create (and maintain) a suitable `.gitignore` file to help you with this.
- Tests and CI: there should be a set of tests (potentially involving mock data; avoid making calls to real APIs in your local tests) to check that basic functionality of your package is not broken. A coverage of 100% is not required (but admirable). These tests should be run in a CI/CD pipeline whenever new commits are pushed to your repository.
- Documentation: there should be dedicated instructions explaining how to install, test, and use your package. Furthermore, there should be as a rudimentary reference manual containing *at least* one-line summaries for all public members (classes, methods, functions) of your package.
- License: choose an appropriate license for your package and publish it (as `LICENSE.md`).
- Contributing guidelines: include contributing guidelines (as `CONTRIBUTING.md`) describing the rules for contributing to your project.
- Every student in your group should have submitted at least one pull request, and submitted at least one review for a pull request of another group member. *Central parts* of your project should

be contributed via reviewed pull requests – but not necessarily all parts have to be. The main branch should be protected, direct commits to it should be disallowed.

- The entire time spent on the project should not exceed approx. 40 hours per student. Submitting a broken project is perhaps not ideal – but not a big deal. This is why the focus in terms of grading is set deliberately on the group reflecting how they can collaborate efficiently. If things go wrong, include a section with the group's thoughts about it in the devlog.
- If generative artificial intelligence is used for any parts of your project, include an explicit declaration in the devlog. Do reflect on your use of AI too: did it provide the expected help in your use cases?

Optional features

The group project is an opportunity to practice other best practices mentioned in the lecture or from external sources. Feel free to experiment with further tools and infrastructure extensions as you see fit. Examples include:

- CI pipelines for linting / formatting code, type checkers
- Prepared (Docker / Singularity / …) containers that can be used to run your image externally
- Shell script installers
- Special pre-commit or pre-push hooks
- A Zenodo-provided DOI for your (publicly) published code
- Storage management frameworks like *Apache Spark* or *HDF5*
- and many, many more…

Implementing additional tools contributes towards the infrastructure points, reflecting about them in the devlog contributes towards the devlog points.

Devlog contents

The following information **needs to be included** in the written report.

- Group members and their responsibilities
- Initial (group approved) project specification
- If artificial intelligence was used: declaration of the use of artificial intelligence
- Comparison with the submitted outcome: in how far were requirements added / removed / modified, and why? In particular, if your submitted project is severely broken: what happened, and how could this have been avoided in hindsight?
- Group collaboration: what rules did you agree upon initially? Was it necessary to introduce changes? If so: which and why? Would a different collaboration model have suited your group more?

With all mandatory items included in your devlog, you can earn up to 12 out of the 15 total points. The remaining 3 points are awarded for any further, optional content[3] that you choose to include in your report (e.g., thoughts on the use of particular tools, additional resources you found helpful, thoughts on how you could monetize your project, …).

[3] Pretty much everything (within reason, and as long as it is related to the development process of your project) can be included; reach out and ask if you are unsure.