

Learning Roadmap for Semester Projects in a World with Generative AI

This document is intended as a guideline for students and supervisors on the first three semesters of the bachelor educations at the Department of Computer Science at Aalborg University. It will be revised in Summer 2027 when we have learned more. It can also be used as a starting point for discussion on all later semesters. The main focus of this document is on semester projects, but some of the advice can easily be applied to courses.

1. Vision, principles, practical advice

Generative AI systems are now part of how software is studied, built, and discussed. You will likely use them during your education, and you will very likely use them in your career. This roadmap is not a list of prohibitions. It starts from a different premise: **You will likely use generative AI in your projects, and what we ask of you is that it is documented, explained, and reflected upon** — in the same way we already ask you to document your methods, your design decisions, and your group process. The central point is that **if you truly understand something**, you can explain it, illustrate it with relevant examples, and communicate it clearly — to each other, to your supervisor during meetings, to the examiners during the examination. This document is a supplement to the universities' general guidelines on [how to use AI in your studies](#). They are more specific and directly applicable to your studies at the Department of Computer Science. Generative AI use is not required, and you or your group can also choose to not use generative AI for specific tasks or for the whole semester as your learning strategy.

Four principles carry everything below:

1. **We value and evaluate learning outcomes over product and report.** Always remember: “The purpose of an examination or test is to assess the extent to which the individual student meets the learning objectives or intended learning outcomes established for the study programme and its components in the curriculum for the specific study programme” [1], i.e. it is your personal knowledge, skills and competences which are assessed. A simple product with understanding of the learning goals is much better than a flashy product with no understanding. Hence, it is not the capabilities of an AI system that are assessed.

2. **Ownership.** Everything you submit and present, you must be able to explain, defend, and modify — **at the exam, without assistance**. If AI produced something you cannot account for, it is not yet yours, and the work is not yet done.
3. **Transparency as a method, not confession.** Whether your group is using AI collectively on a regular basis or only by some members for specific tasks, *documenting how you use AI throughout the project is an important part of the process analysis*. Your AI interactions — what you asked, what you got, what you kept, what you rejected and why — must be documented regularly, to a degree of detail that allows you to reflect and learn how to use AI better in your studies and your career.
4. **Protect the skill under construction.** Semester projects build specific learning capabilities and software skills in you. Generative AI can support the capabilities you already have and help you strengthen the ones you are building — but if it **performs** the skill the project is designed to train, you graduate with a gap where a competence should be [2,3].

These principles are in alignment with the [general PBL principles of AAU](#).

Practical advice:

- Before delegating a task to AI, ask yourself: *"Is this the central skill I am here to learn, or is it a peripheral skill or concept?"* E.g, a piece of code in C is a central concept in P1, making a graph look nice could be a peripheral skill.
- Keep an AI-log/journal. Keep it lightweight and honest; a personal journal that feeds directly into your process analysis. Share it regularly with your supervisor.
- An individual entry in the AI-log/journal could consist of the following items
 - which model did you use?
 - in relation to which part of the project (e.g. coding, writing, related work)?
 - what kind of tasks the prompt had? (e.g. question, generate file, ...)
 - did you use the answer or spend some time sparring/refining?
- At the exam, you will be asked about anything you submitted, AI-assisted or not.
- When unsure whether a use is appropriate: do it, log it, and raise it with your supervisor. Discussable beats hidden, every time. (Don't share sensitive data with external AI tools)
- Try to have a mid-term meeting with your supervisor where each member of your group practices explaining and communicating the problem, the research questions, the ongoing development work, etc. No need to make slides — focus on practicing the skill of communicating your understanding. This resembles the questioning that you will experience during the exam.

2. AI can be useful in many different ways

Semester projects, especially PBL ones at Aalborg University, involve not just learning or programming, but other activities like formulating the problem and research questions, conducting related work, writing and documenting, testing and making software outputs robust and reliable. Below is a suggestive list of how AI can be useful in some of the activities of your projects, as long as you follow the four principles outlined above:

- **Learning & understanding** — using AI as a tutor, explainer, question-answerer, quiz partner. One potential tool is the Socratic PBL-Agent listed on the [page about AI as a study tool](#).
- **Investigating the problem** — search, literature, domain understanding, stakeholder perspectives.
- **Programming** — code generation, debugging, refactoring, testing.
- **Writing** — report prose, structure, language polish (many of you write in a second language). Remember that writing is thinking.
- **Documenting & reflecting** — the process analysis, the AI-log itself, learning journals.
- **Planning & collaborating** — time plans, meeting agendas, task breakdowns, group coordination.

The sections below make explicit for each semester project what is its main skill that is "under construction" and which therefore needs your own hands on it first.

3. Using AI in Semesters Projects

P0 — Analysis & Problem Formulation

Skill under construction: analyzing a real-world situation and formulating a problem **without** jumping to a solution. This is the heart of PBL and the hardest habit to build — precisely because AI tools are solution machines. Ask an AI about a problem and it will hand you five solutions before you have understood the problem.

You are free to use AI during the writing and exploration of the problem domain. You should critically reflect on your use of AI in the project in a separate section. Whenever an AI offers you a solution, write it down in your log — and then set it aside. Solutions come later.

- **Learning:** Freely — ask AI to explain PBL, problem analysis, the difference between initiating problem and problem statement.

- **Investigating:** Use AI to map the terrain (who is affected? what is already known?), but verify against real sources, and talk to real people where possible.
- **Writing:** Draft your problem formulation first. Then, if you wish, ask AI to critique it — is it too broad, does it smuggle in a solution? — and check style, clarity, and language. **Do not ask AI to write it.** Whatever role AI played in your text, however small, note it in your log: this is where the habit starts. You do not need to record all interactions with AI, but you need to log enough of your interactions to reflect on and learn from your practice.
- **Documenting & Reflecting:** Start your AI log now. One page: what you used AI for, one moment it helped, one moment it misled you.
- **Planning:** Freely — agendas, timelines.
- **Programming:** Not required, but optional. Be aware that this could waste a lot of your time.

P1 — A program that solves a problem

Skills under construction: programming and reasoning about a problem.

Programming is memory, pointers, control flow, debugging, and reasoning. If AI writes your C, you will pass nothing and learn less. This is the project where "hands on the core" matters most for code. Reasoning about the problem involves attacking the problem from many angles, constructive discussion, brainstorming, and much more.

- **Learning:** Freely and enthusiastically — "explain segmentation faults," "quiz me on pointer arithmetic," "why does this compile but crash?" AI can be an outstanding, infinitely patient C tutor, but only if you use it in this way. Don't use AI just to solve exercises — don't cheat yourself.
- **Programming:** Write code yourself first. Use AI to **explain** errors, review your code, and suggest test cases — not to generate solutions. A good rule: AI may read your code; it should rarely, if at all, write it. When it does write something (e.g., boilerplate), mark it, understand it line by line, and log it. If you ask AI to write code for you, it can be good to ask it to generate explanatory comments alongside the code.
- **Version control & collaboration:** Freely — AI as a Git tutor ("explain rebase," "help me untangle this merge conflict step by step") is one of its best uses, and the tool-fluency it builds is exactly what the module asks for.
- **Writing:** You draft first; AI as critic after — for structure, clarity, style, and language polish (many of you write in a second language, and this is a legitimate and valuable use). Always check and read every reference AI mentions before citing it, it might not exist or say the complete opposite of what you want to cite. Log what you asked, what changed, and what you rejected: AI-assisted writing is documented the same way AI-assisted code is.

- **Documenting & reflecting:** Your process analysis should now include a section on AI: where did it accelerate you, where did it tempt you to skip understanding?
- **Planning:** Freely, but be mindful of your learning and not outsourcing your thinking to AI.

P2 — A larger program developed by a group

Skill under construction: the step from **writing code** to **designing a program** — decomposing a problem into a model, choosing data structures and algorithms deliberately, and building something larger than any one group member holds in their head, possibly across a network. P2 is explicitly the semester where you build the common programming foundation that every later semester stands on. Therefore, if you use generative AI in P2, the goal is to use it in ways that strengthen that foundation rather than substitute for it: as an explainer of algorithmic concepts, a critic of your designs, a reviewer of your code, and a partner in testing — while keeping the design decisions, the core implementation, and the reasons behind them in your own hands. Learning to draw this line, and to document where you drew it and why, is itself one of the professional skills of P2.

- **Learning:** Freely — AI is excellent at explaining algorithmic concepts, complexity trade-offs, network protocols, and unit-testing ideas. "Why would a hash map beat a list here?" is exactly the right kind of question.
- **Investigating the problem:** As in P0/P1 — AI for mapping the terrain; your own delineation and defense of the problem. The problem formulation remains yours.
- **Programming:** This is the central activity. **Design and model first, by hand and on the whiteboard, as a group** — the model, the module decomposition, the choice of algorithms and data structures are the examinable core, and they must be yours. Once the design is yours, AI may act as **critic and reviewer**: propose your structure and let it attack; ask it to review a function you wrote; ask it to suggest edge cases for your tests, where you might need to reconsider what the correct result should be. AI-generated code should remain the exception, confined to genuinely peripheral boilerplate — and everything kept must be understood line by line, marked, and logged. A concrete test: if AI wrote it, could you re-derive it at the oral exam with the tool switched off?
- **Testing:** Writing tests is a learning objective this semester, not a chore — so write your unit tests yourself first. AI may then propose **additional** cases you missed; log which ones survived and why. Do not delegate the whole test suite: deciding **what to test** is the skill. You need to develop the skill of thinking like a tester and not just a developer.

- **Writing:** This is the semester where you might use AI-assisted writing as a deliberate practice: how to make a text your own by rewriting and restructuring, how to instruct AI toward a style, how to write short and concise with AI while keeping your own point. As always: check every reference yourself, and document the collaboration in your log.
- **Documenting & reflecting:** Your process analysis could now track AI at the **group** level: did AI use diverge between members? Did it create knowledge of asymmetries in the group — code one person "owns" but nobody understands? That is a genuine project-management finding, and it connects directly to the module's reflection objectives.
- **Planning & managing:** Freely — sprint plans, task boards, meeting agendas, status-seminar preparation.

P3 — A well-structured application

Skill under construction: design choices and judgment across a whole development process — from requirements modeling through architecture and design patterns to systematic testing and evaluation of the user interface. The examinable core of P3 is that you must be able to **argue for the choices made in every activity of the process**. AI can produce plausible OO designs, pattern applications, and architecture diagrams instantly — but it cannot argue and justify **your** choices, because arguing a choice means owning the **alternatives you rejected and why**. A design you cannot defend at the exam is not your design, whoever typed it.

- **Learning:** Freely — design patterns, architectural styles, OO analysis concepts, integration-testing ideas. Asking AI to compare two patterns for your situation is exactly the right kind of question.
- **Requirements & modeling:** AI can help you practice — role-play a skeptical stakeholder, stress-test your use cases, question your domain model. But requirements come from the problem and its users, not from an AI. Design the high-level view, the specifications, the abstract model first; let AI interrogate the model after.
- **Programming:** You are now expected to have the foundation from the previous two semesters, and AI-assisted implementation of well-specified classes is a legitimate professional workflow worth learning. The trade: every accepted piece of code must fit **your** architecture, and the architecture, the design pattern choices, and their rationale must be yours and must appear, argued, in the report. "The AI suggested Observer here" is not an argument; why Observer beats the alternatives **in your context** is.
- **Testing:** Letting AI generate unit tests for your classes is now acceptable and worth practicing. But P3's distinctive objective is **integration testing** — showing

how the parts work against user needs. The test strategy (what to integrate, in what order, etc) is design judgment; keep it yours, let AI critique it.

- **Evaluating the user interface:** UI evaluation means real users. AI can help you design the evaluation — draft tasks, sharpen questions, pilot your protocol — and later help analyze what users say. It cannot **be** the users. Synthetic "user feedback" generated by an AI is not evaluation data, and a report built on it has skipped the learning objective entirely.
- **Writing:** Continue developing the AI-assisted writing practice from P1/P2 — and note that in P3 the report's **argumentation** is itself the assessed skill, so a generated argument doesn't just weaken your learning; it removes the very aspect being examined and you are expected to learn as a skill.
- **Documenting & reflecting:** Documenting your interaction with AI becomes even more crucial, because it involves both your creative and critical thinking about software design. You could include a documented **design dialogue** — your proposed architecture, AI's critique, your defense or revision. By P3 this should be less like an exercise and more like how you actually work: the dialogue is your evidence that AI sharpened your judgment rather than replaced it.
- **Planning & managing:** Freely.

References

- [1] Translation from “Bekendtgørelse om eksamen og censur ved universitetsuddannelser “ <https://www.retsinformation.dk/eli/lta/2020/22>
- [2] Bastani, H., Bastani, O., Sungu, A., Ge, H., Kabakcı, Ö., & Mariman, R. (2025). Generative AI without guardrails can harm learning: Evidence from high school mathematics. *Proceedings of the National Academy of Sciences*, 122(26), e2422633122.
- [3] Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., ... & Briggs, B. (2024, August). The widening gap: The benefits and harms of generative ai for novice programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1* (pp. 469-486).