# Criterion A: Planning

## The scenario

My client is the **CAS Coordinator at my school** and the adviser is **my Computer Science teacher**.

Being a CAS coordinator, my client manages student's portfolios which are websites (based on Google Site) used to post reflections and evidence for each CAS strand – Creativity, Activity and Service. While attending her classes, I noticed that she used Excel files, one file for one class, to store links to students' portfolios (our school doesn't have Managebac or any similar service). I found her approach rather problematic and arranged a meeting with her (Appendix A.1) to receive feedback on my ideas how to solve this issue.

After a short conversation, I proposed a program, **CAS Manager**, that would enable the storage and retrieval of students' portfolios from a single convenient database. Furthermore, CAS Manager would simplify the management of students' portfolios even more by providing a useful GUI with a set of tools for displaying, altering, and deleting students and, as a result, would make the process much more user-friendly.

I arranged a second meeting with my client (Appendix A.2) to determine the precise functionalities of the program, according to the needs of my client. After this meeting, I was able to establish precise success criteria. Later, I also scheduled a third, last meeting, before starting the development of the program, to get the client's final feedback and ensure that the success criteria meet her needs (Appendix A.3).

## Rationale for the proposed solution

For developing the app, I opted for Python due to its support for Object-Oriented Programming (OOP), similar to Java. This OOP approach is beneficial for managing transactions with student data in the database, where each student has associated attributes (first name, surname, class, portfolio URL). Using classes allows for storing this data and defining relevant methods for transactions.

Python's multi-paradigm nature also supports functional programming, aiding in faster test writing with libraries like pytest. Functional design avoids the boilerplate code often associated with OOP in Java, streamlining the testing process.

Despite Python's comparative slowness to languages like C++, its elegant syntax is advantageous for delivering a better user interface experience. This aspect is crucial as the GUI plays a central role in my application.

**Success criteria**

1. Add new and update existing students
2. Add new and update existing classes
3. Open one or more students' portfolios
4. Choose either descending or ascending alphabetical ordering of students that are displayed
5. Sort students by either surname or first name
6. Restrict student are displayed to a chosen class
7. Search for students by their surname case-insensitively and without the need to type the whole surname
8. Delete individual and multiple students' records as well as entire classes
9. Make the app resizable
10. Enable the change of theme – either black or white
11. Validate data before insertion

**Word count:** 343