

Analysing & Solving Problems
ACCA4019

Component 1-
Coursework: Portfolio

Submitted by
Fernando Bruno Peternella Moreira

Student number:
2523396

Date: 19 July 2026
UWSTD – Birmingham University

Table of Contents

- Table of Contents..... 2
- Introduction 3
- 1. Problem Deconstruction 4
 - 1.1 Requirements and Constraints 4
 - 1.2 MoSCoW Analysis..... 4
 - 1.3 Key Variables..... 5
- 2. Proposed Solution 6
 - 2.1 Flowchart 6
 - 2.2 Explanation..... 7
 - Worked Example..... 7
 - Assumptions 8
 - 2.3 Justification..... 8
- 3. Reflection and Justification 9
 - 3.1 Techniques Used 9
 - 3.2 Critical Evaluation..... 9
- References..... 10

Introduction

This report is about a problem that many students have in college: it is not easy to organise a meeting for a group project. Students have different timetables, part-time jobs and personal things to do, so it is hard to find a time when everyone can meet. Because of this, some groups do not finish the work on time, and the tasks are not divided in a fair way between the members.

This report will break the problem into small parts, using tools like MoSCoW and an IPO chart. After that, a solution is proposed, with a flowchart to help groups find good meeting times. A small worked example is also included, to show the logic working with real numbers. At the end, the techniques used to solve the problem are discussed, together with the limits of the solution.

1. Problem Deconstruction

1.1 Requirements and Constraints

A requirement is something the solution must do to work. A constraint is a rule or limit that the solution must respect, like a deadline, a budget, or a technical rule [6]. The table below separates these two things for the meeting-scheduling problem.

Requirement	Constraint
Collect the free time of every group member.	Must be free to use — no paid software or app.
Find a time that works for the group.	The meeting must happen before the project deadline.
Avoid times that clash with a class or a job.	It must be simple, so no coding skill is needed.
Tell the group when there is no good time for everyone.	The process cannot take a long time to decide.
Give another option when not everyone can join.	It must work every week, not only one time.

1.2 MoSCoW Analysis

MoSCoW is a simple way to put the features of a solution in order of importance: Must have, Should have, Could have, and Won't have [1]. It was used here because not every feature is equally important — some are needed for the solution to work at all, and others are only nice extras.

Category	Feature
Must have	Collect everyone's free time; check if a slot is possible before the deadline; warn the group about conflicts.
Should have	If 100% is not possible, choose the slot where most people are free.
Could have	Send an automatic reminder about the chosen time.
Won't have (this version)	Connect automatically with Outlook or Google Calendar — this is too complex for now.

1.3 Key Variables

The problem can also be organised with an IPO chart (Input, Process, Output), the same tool used before in class for the Contractors' Pay Calculator exercise. Writing the problem as Input/Process/Output helps to see exactly what information is needed before any decision is made.

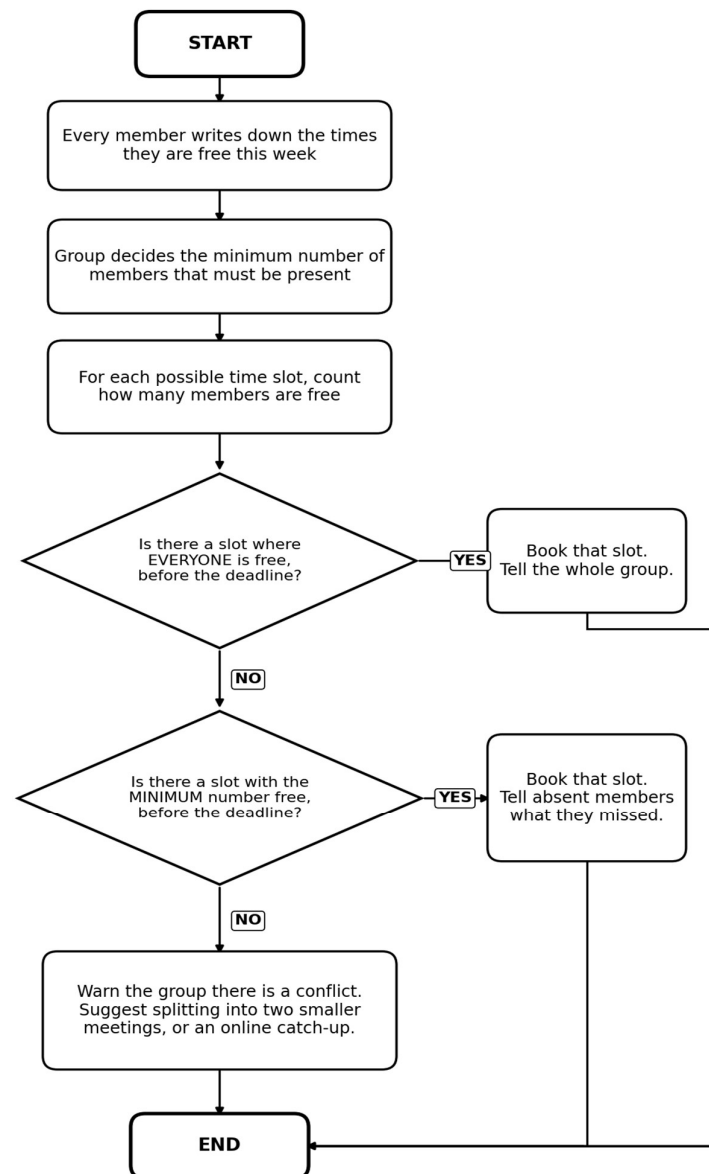
Input	Process	Output
Name of each group member	Compare the free time of each person	The meeting time
Free time slots of each member	Count how many people are free in each slot	List of who can attend
Project deadline	Check if the slot happens before the deadline	Conflict warning (yes/no)
Meeting length; minimum number of people needed (group decides)	Choose the best slot, or suggest another option	Alternative suggestion, if needed

These variables matter because they are the information the group really needs to make a decision. Without knowing who is free, the group cannot compare timetables. Without knowing the deadline, the group cannot check if a slot is valid. And without a minimum number, the process could get stuck forever, waiting for a moment when literally everyone is free — which, in a real student group, may never happen.

2. Proposed Solution

2.1 Flowchart

The flowchart below shows the full process, from collecting free time to booking a meeting, including what happens when a conflict is found.



2.2 Explanation

The flowchart solves the problem in a few simple steps:

First, finding a time for everyone. The process looks at the free time of each member and counts how many people are free in each slot of the week. This is the same idea used in the trace table exercises from class (Counter, Swap): go step by step through each slot, and update the count each time.

Second, reducing conflicts. The scenario says students have jobs and different commitments, so it is normal that not every slot has 100% of people free. Because of this, the group decides its own minimum number that is acceptable (for example, 3 out of 4 members, or a different number, depending on the group). This way, the process does not get stuck waiting for something that is almost impossible.

Third, meeting before the deadline. Every time the process picks a slot, it checks if that slot happens before the project deadline. If it does not, that slot cannot be used, no matter how many people are free.

When there is a conflict — no slot with the minimum number of people, before the deadline — the flowchart does not just stop. It suggests an alternative, like splitting the group into two smaller meetings, or doing part of the discussion online. This way, the group still moves forward instead of losing the whole week.

Worked Example

To check the logic works, a small example was traced by hand, using 4 members (Ana, Ben, Cara and Dan) and a deadline of Thursday 17:00. The group decided that a minimum of 3 members is acceptable.

Slot	Ana	Ben	Cara	Dan	Count
Mon 18:00	Yes	No	Yes	Yes	3
Tue 12:00	Yes	Yes	No	Yes	3
Wed 18:00	No	Yes	Yes	Yes	3
Thu 12:00	Yes	Yes	Yes	No	3

Following the flowchart: Decision 1 asks if any slot has all 4 members free. Looking at the table, no slot has a count of 4, so the answer is NO. Decision 2 asks if any slot has at least the minimum (3) free, before the deadline. Monday 18:00 has 3 members free and happens before Thursday 17:00, so the answer is YES. The flowchart then books Monday 18:00, and tells Ben — the one member who is not free — what he missed. This small trace shows the same step-by-step checking used in the Counter and Swap exercises, but applied to a real scheduling decision.

Assumptions

- Every member has internet access to fill in a simple form (for example, Google Forms).
- Members update their free time every week, and in an honest way.
- One person in the group is responsible for running this process; it does not need to be automatic.
- Time is divided into fixed blocks (for example, one-hour blocks), to make the comparison easier.

2.3 Justification

This solution was chosen because it is simple and free — any student can follow the steps without a special program or app, which matches the constraints from Section 1.1. The most important part is the minimum number decided by the group: instead of always needing 100% of members (something that almost never happens in real life), the process accepts a solution that is good enough. This idea is similar to classic scheduling problems in computer science, where a simple rule is used to choose activities or time slots so that nothing overlaps [5].

Compared to the normal way groups try to organise meetings — messages back and forth in a group chat, with no clear rule for when to stop trying — this flowchart gives a clear, repeatable process. Every week, the group can follow the same steps and reach a decision quickly, instead of losing days waiting for everyone to reply.

3. Reflection and Justification

3.1 Techniques Used

Abstract thinking: a real, messy problem (lost messages, busy people, clashing timetables) was reduced to a few important variables: who is free, when, and what is the deadline. Small details that do not change the decision, like the reason why someone is busy, were left out on purpose. This is the same thing done in the Contractors' Pay Calculator exercise — take a business problem and reduce it to a small number of variables [4].

Subdividing: the big problem was separated into small steps: (1) collect availability, (2) compare time slots, (3) check the deadline, (4) decide and inform the group. This follows the idea of breaking a complex problem into smaller, easier tasks, instead of trying to solve everything at once [2].

Systems thinking: the group of students was treated like a system, with an input (schedules), a process (comparison), and an output (a booked meeting or a conflict warning). Changing one part of the system, like the minimum number, changes the final result for the whole group [3].

Trace table: in the same way the Counter and Swap exercises were checked step by step, this flowchart was tested with a small example (4 members), shown in Section 2.2, to see if the logic gives the correct answer before it is used for real.

3.2 Critical Evaluation

This solution is simple and low-cost, which is good for students. But it has some limits. First, it only works if members update their availability correctly — if someone forgets, the whole process fails. Second, it does not deal well with last-minute changes, for example when someone is called into work suddenly; the flowchart would need to be run again. Third, the fixed time blocks may not capture small windows of free time, like 20 minutes, so some possible slots could be missed. Fourth, the process is manual, so for bigger groups (6 or more people), comparing schedules by hand becomes slow and difficult, and mistakes are more likely.

References

- [1] Business Consortium, Agile. "What Is MoSCoW Prioritization? - Agile Business Consortium." Agile Business Consortium, 28 May 2026, <https://www.agilebusiness.org/resource/what-is-moscow-prioritization>. Accessed 19 July 2026.
- [2] GeeksforGeeks. "What Is Problem Decomposition? - GeeksforGeeks." GeeksforGeeks, 23 July 2025, <https://www.geeksforgeeks.org/what-is-problem-decomposition/>. Accessed 19 July 2026.
- [3] Lutkevich, Ben. "What Is Systems Thinking? | Definition From TechTarget." TechTarget, 15 May 2025, <https://www.techtarget.com/searchcio/definition/systems-thinking>. Accessed 19 July 2026.
- [4] Seward, Kim. "Intermediate Algebra Tutorial 8: Introduction to Problem Solving." West Texas A&M University Virtual Math Lab, 1 July 2011, https://www.wtamu.edu/academic/anns/mps/math/mathlab/int_algebra/int_alg_tut8_probsol.htm. Accessed 19 July 2026.
- [5] GeeksforGeeks. "Activity Selection - GeeksforGeeks." GeeksforGeeks, 8 Sept. 2025, <https://www.geeksforgeeks.org/dsa/activity-selection-problem-greedy-algo-1/>. Accessed 19 July 2026.
- [6] Central Connecticut State University. "CS 410/510 - Software Engineering." Central Connecticut State University, <https://cs.ccsu.edu/~stan/classes/CS410/notes16/04-Requirements.html>. Accessed 19 July 2026.