Guided Project, Part 1.

The project will be ideally implemented using the IntelliJ or Eclipse environment: Java with a Maven project and JUnit for unit tests. You will use Git and Gitlab (https://gitlab.esiea.fr with the Azure AD login) and work in pairs.
You will work in pairs or in trios, but each member of the team is expected to create minimum 3 use cases at the end of the project, on different branches of your project; branches will be merged at the end and so you will deliver minimum 6 working or 9 working if in trios, tested use cases in one program.
Beware that you are not allowed to write code until STEP 12. With STEP 1 to STEP 11 you are DESIGNING your application. Starting with STEP 12, you are IMPLEMENTING it. During the implementation, you can update your design.
All the diagrams (use case, class, package, sequence, component) and the use case description tables will be committed in git as well, in a folder named docs.

In this part, you will describe the use cases for the application of your choice. Validate your choice with the teacher before going on.

STEP 1. Write Use Cases

Follow this method:

1. Work top-down: Actors, Goals, Main story, Alternative conditions, Alternative paths.

2. Brainstorm the actors, human and non, who have an operational goal against the system.

3. Brainstorm their goals against the system.

4. Draw the UML Use Case diagrams of your actor-goal items. To this purpose, you might want to use https://www.draw.io/

5. Write the use cases. Use the table given below: fill it in

6. Write how the interests of the Actors are satisfied at the successful conclusion of the use case.

7. Write what interests must be protected in case there is failure of the use case.

8. Write the precondition: what the system has ensured is certainly true.

9. Write what event or thought triggers the main success scenario.

10. Write the main success scenario.

11. Write each sentence as a goal succeeding, distinctly moving the process forward. Show the intent of the Actor, what it/they want and get accomplished in that step. Avoid user interface descriptions.

12. Manage the level of goal accomplishment so that the scenario is between 3 and 5 steps long.

13. Check that the sequencing requirements in the steps - or lack thereof - are clear.

14. Brainstorm the failures and alternative paths.

15. Include only the failures the system must detect and handle.

16. Write the failure or alternative condition as a condition phrase or sentence.

17. Write a scenario fragment showing how the alternative course leads to goal failure or success.

18. Write using the same rules as for the main success scenario.

19. Update the main success scenario with the new validations found while writing the extensions.

Review and agree with your pair on the use case description.

Use case (name)		
Pre-conditions		
Success end condition		
Failed end condition		
Actors		
Trigger		
DESCRIPTION	Step	Action
	1	
	2	
	3	
	4	
	5	
Extensions	Step	Branching action
	1	
	2	
	3	
Sub-variations	Step	Branching action
	1	
	2	
	3	
Performance		
Frequency		
Open issues		