COMP1035 – Lab 03: UML Diagrams I

Today's Task:

1. Create Activity Diagram(s) & Sequence Diagrams.
2. Collaboratively write a markdown document about these diagrams and link it to README.md.
3. Tag your repository with a meaningful tag message upon completion.

Team Advice:

1. Consider dividing into two **sub-teams**: each sub-team creates an activity diagram. Gather along afterward to discuss and refine. Repeat the process for sequence diagrams. Determine if working in pairs or sub-teams is more efficient and **assign responsibility for a member to bring everything together on time**.
2. Suggest one group member focuses on mastering the software while others begin sketching UML diagrams on paper. This separation of tasks aligns with common software engineering practices. Share sketches with group members and incorporate them into Visual Paradigm as needed.
3. Feel free (should) to add explanatory notes to clarify diagram choices and highlight important issues or questions for further discussion (with stakeholders).
4. Refer to the Visual Paradigm User's Guide for assistance if required.

Reminder: Software Brief

The university wants a new piece of software for handling module options and allowing students to sign up to optional modules. This includes students from other departments that might want to take the first year. Now, students must collect a form from student services, optionally attend an introduction lecture to a few options, write their choices on the form, and return it to student services. These choices are then checked and, by default, approved if they add up to 120 credits, evenly with a 60-60 credit split each semester, and all from their school and at the right year level. There are many cases, however, where additional approval is required. Some modules require approval from the module convenor, if for example, the student must have taken other pre-requisite optional modules before it. Also, if a student wishes to take a 50-70 credit split, they need approval from the Head of Teaching; this is usually dependent on how well the student is doing and whether they are likely going to be able to handle 70 credits in one semester. Further, if a student wishes to take an introductory module from another department, such as first year Japanese or Introduction to Economics, then the student needs approval from a) the module convenor of that module, and b) from the Head of Teaching in their own school. Approval from the module convenor is often based on limited class sizes, and on deciding whether the student has the pre-requisite learning necessary to understand the classes. This prevents a student from one school taking an advanced subject outside of their discipline, with the likelihood of failing it. Students wishing to take more advanced level foreign languages, for example, may need to provide evidence of language

ability if they learned the language outside of the university. The student must gain all these approvals before submitting their forms to student services. These choices must be fed into other university software, two of which are: BlueCastle (student records) and timetabling (for room sizes).

A. Activity Diagram & Sequence Diagram(s) [60 to 80 mins]

Identify two **aspects** of the software suitable for representation by either an activity diagram or sequence diagram. In step B, provide explanations for each aspect: a) specify the purpose of building the diagram, and b) justify why this diagram type is most suitable.

1. Activity Diagram

Choose a complex process involving decision points and ideally, concurrent activities. Create an activity diagram to model this process. You should try to aim for modelling complex scenario (instead of simple one) and incorporate splits and joins (features) where applicable.

2. Sequence Diagram

Select a process characterised by a sequence of interactions between objects or individuals. Develop a sequence diagram illustrating the sequential flow of these interactions, detailing the sender(s), recipient(s), and order of information exchange. There are many aspects of boxes/arrows in sequence diagrams that are confusing and yet to be thought. You may choose to ignore these (if you want to learn more, please refer to the sample brief and solution) for now and focus on what you are trying to convey in the diagram.

B. Create Markdown(s) for Task A in Team's Repository [30 mins]

1. You can see a placeholder markdown file in the **"/docs/" folder**, of the git repository. You can use this, but you can also create **additional markdowns** that this refers to. This is entirely up to the team's decision.
    - You should make images of your diagrams above and put them in either the "/images/" folder of the group repository, or anywhere else you think is more appropriate (like inside a Lab-03 subfolder of "/docs/" that's dedicated to this report).
        - You can export those images (save as image), or screenshot them, whatever gets you an image of them. The main thing is to make sure they are **readable and clear**!
    - In the Markdown(s), make sure you have the following contents clearly visible:
        - What **aspect of the brief** do you think would benefit from a diagram/model?
        - Which **type of diagram** do you pick for this and **why**?

    - **NOTE**: **Do not give bookwork answers** (e.g., activity diagrams are good for X, Y and Z) – focus on why the aspect you want to convey is suited to this type of diagram, and why not other types.

2. You should add a link to this markdown in README.md.
    - Bryan should be able to access the link in the README.md to your group's markdown from this week and access/view all the markdowns without browsing the folders in the group repository.

3. Your report should have:
    - A title.
    - A very brief introduction to the report (Lab-03).
    - For each diagram
        - A reasoning for why it is worth producing.
        - An image of the diagram.
        - Key notes about the diagram, or questions that you have about the software based on what the diagram makes you realise you need clarification on.

4. (Reminder) Lastly, **do not forget to link this markdown to the main README.md**.

C. Tag Your Repository [5 mins]

Use the git tag process to tag your repository. This should be the last thing you do. You should use a tag name, e.g., "Lab-03-v*X.X*" and tag message, e.g., "Lab03 UML diagram markdown report version *X.X*" so that it is clear which version of your repository I should refer to). Remember, this creates a snapshot-point of your group repository that I can go back at any time and look at. If you have multiple versions, remember to have a good **version history documented in the README.md**.

D. IMPORTANT: Peer-Review Your Team [5 mins]

Before **12 noon today**, provide the peers' review of your groupmates for their contribution to this week's lab work. Everyone **SHOULD** do this for Lab 03 and think carefully about what you are giving yourself and every person in the team.
