COMP1035 – Lab 03: UML Diagrams I

Today's Task:

1. Fix up the team's repository if have any feedback from Lab-02.
2. Make Activity Diagram & Sequence Diagrams.
3. Produce markdown (as a team) about these diagrams.
4. Tag your repository at the end with a clear tag.

Team Advice:

1. You might consider dividing up into two sub-teams – both sub-teams make an activity diagram. Later, gather along and discuss. Both sub-teams make a sequence diagram – come back together and discuss. In general, consider whether you are working in pairs, sub-teams, and **who's in charge of bringing it together on time**.
2. I recommended that one person in e.g., a pair gets to grip with software, while the others begin to sketch UML diagrams on paper. This separates reasoning from constructing – something that's common throughout software engineering (SE). It's also hard to bring diagrams together into 1 project. When 'finished' with sketching the diagram, take a photo, put it in your team chat room, give it the people using visual paradigm, and then e.g., move on to the next diagram.
3. You can (*should*) add notes to explain diagram choices if that helps make them clearer. As well as notes about important issues, and questions to find out more.
4. Visual Paradigm User's Guide.

Reminder: Our 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 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]

You should identify two **aspects** of the software that can be captured by either an activity diagram or sequence diagram. In step B, you must explain before each one a) what you are building a diagram for, and b) why this type of diagram is the best for it.

1. Activity Diagram

Find a process that is complicated and involves decisions and ideally, parallel activities and create an Activity Diagram for it. You should try to model something complicated, rather than simple, and try to make use of splits and joins where possible.

2. Sequence Diagram

Find a process that has a series of communications between objects or people and create a Sequence Diagram explaining the order that they must happen, and who/what send which information and in what order. There are many aspects of boxes/arrows in Sequence Diagrams that are confusing and yet to be thought. You should ignore these (unless you want to learn more) 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 git 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 them, 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 report (or the front page of the report if it's a multi-file report) in the root README.md file.

- Bryan should be able to open the README.md file and find a single link to your team’s markdown from this week and view all the markdown (if multi-line) without browsing the 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. 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 team repository that I can go back at any time and look at. If you have multiple versions, remember to have good **version history documented**.

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

Before **2 pm today**, go to Moodle and peers’ review your teammates 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.
