COMP1035 – Lab 02: Text Analysis and Use Case Diagrams

Team Tasks: (Work in Team)

1. Identify Stakeholders with Text Analysis.
2. Try a Use Case Diagram in Visual Paradigm.
3. Export and Insert the Diagrams in Markdown (as Figure) in Team Repository.
4. (Optional) Visual Studio Code – a free IDE development operation such as debugging, task running and version control which supports Git and Markdown.

Get Started with the Software

1. Install Visual Paradigm Community Edition on your own laptop.
2. Get a **free activation code with university email** and activate the Visual Paradigm.
3. Start opening Visual Paradigm! For the first time, enter the activation code you received.
4. Start looking at the Visual Paradigm guides for Text Analysis and Use Case Diagrams.

Important Notes:

Your team is in a software company. A client approaches your company for a new project. You probably have an email enquiry, that says a little something about what they want, and your team schedule a meeting to talk about it with your client.

You want to prepare some initial thoughts, that you can use to have a productive first meeting.

- You have thought about the types of users, so you can discuss them.
- You have thought about the kind of use cases, so you can discuss them.
- You have a list of questions to ask them to clarify things.

Then after the meeting, you can define the requirements better. To make good content, consider of the following:

- What about doing a good stakeholder analysis? (Answer to this can go in Markdown)
- Makes clear textual analysis.
- Makes a good use case diagram. (More advanced features?)

NOTES:

- Producing basic diagrams is like 40% of the way there.
- Producing good diagrams that are useful is 70% the way there.
- Explaining reasoning for choices and useful insights is the rest.

The 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. Text Analysis in Visual Paradigm [30 mins]

You should discuss this with your team, (for international students - perhaps someone screen sharing), as you do it. You can copy and paste the brief into **Textual Analysis Look** for types of user (stakeholder), and tasks/activities (use cases) that people can do with the software. There are other things you can identify in a Textual Analysis – but we will get to that in later classes/labs. Refer to Visual Paradigm guides to learn about doing it.

Suggestions and Tips:

1. You can give friendly names to them.
2. You can colour code for categorisation.
3. You can add notes.
4. Do a stakeholder analysis and add to your markdown.

B. Use Case Diagram [60 mins]

In a Use Case Diagram, you create a 'use case' for all the main tasks that different types of users can do. If multiple types of users can take part in the same use case, then they can both be connected to it. Some activities might require other activities (like logging in), which can be connected. **You can *should* add notes to explain things if that helps make them clearer**. Refer to Visual Paradigm guides to learn about doing it.

If you want to learn more about Use Case Diagrams, such as

1. Showing different types of connections,
2. Showing special cases of use cases or
3. Bounding different system boxes of use cases,

You can do independent investigation into more advance features. Useful links are in Moodle for Use Case Diagram etc. and many more resources exist online.

Tips:

1. Use the use case button – not the activity button (feel free to do research into why)
2. Some buttons have a little black arrow in the bottom right corner – this lets you choose alternative, e.g., lines to draw.
3. Use Case Diagrams – are not Flow Diagrams.

C. Export and Put the Content in the Team Repository [30 mins]

1. Export images (or screenshot – whichever given it must be readable!) of your visual paradigm work and put them in the assets folder of your team repository. You should **NOT** be putting visual project files into the team repository.
2. Together, create one or more markdown pages about your work, where you can put explanatory text from the two tasks above around the diagrams. Embed the diagrams into the markdown, so you can see them.
3. Link this markdown from the main README.md, so Bryan can find it.
4. At the end, ONE person tags the repository for the work you achieved by typing:

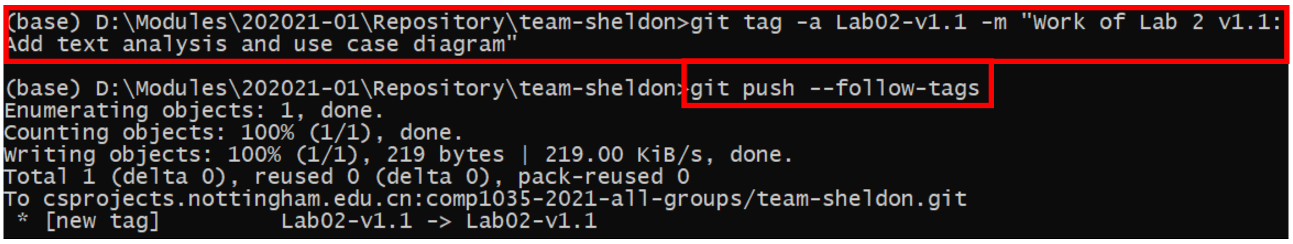
   ```
   git tag -a <tag> -m <description>
   ```
   E.g., `git tag -a Lab02-v1.1 -m "Lab 02 version 1.1"`

   Then, make sure the tag is pushed to the server by typing:

   ```
   git push --follow-tags
   ```

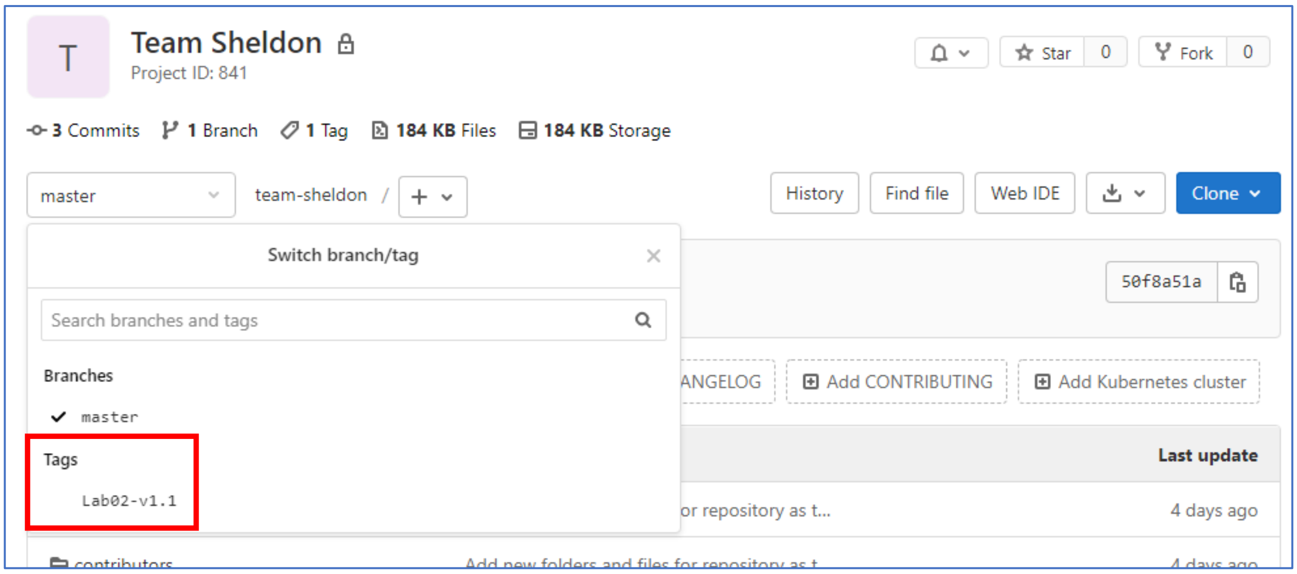
5. Think from Bryan's perspective that he can:
   a. Find the tag in the menu on the left,
   b. View the README and see a good documentation of version logs, a good SAMPLE from Google Chrome. May create a new Markdown, e.g., Logs to store all version history (remember to include a link in README).
   c. Use it to follow your Lab's work.

Sample Output:



```
(base) D:\Modules\202021-01\Repository\team-sheldon>git tag -a Lab02-v1.1 -m "Work of Lab 2 v1.1:
Add text analysis and use case diagram"

(base) D:\Modules\202021-01\Repository\team-sheldon>git push --follow-tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 219 bytes | 219.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To csprojects.nottingham.edu.cn:comp1035-2021-all-groups/team-sheldon.git
 * [new tag]         Lab02-v1.1 -> Lab02-v1.1
```



Refer to git tag documentation HERE for more details.

D. Peers Assessment [5 mins]

The link is provided in Moodle, under Week 02 – Lab. Rate how well every member, INCLUDING yourself contributed to the work this week. If you did not complete the peers' assessment, assumably you rate "Excellent" for all your team members.

Time: Today 10.30 am to 2 pm.
