ENGR 301 Assignment 1

Draft: submission is *likely* to be via Nuku; please be patient while this is finalised. The questions below will only be modified if there are errors, omissions or clarification required. Please feel free to start composing answers at your earliest convenience.

Project Management Body of Knowledge

[35% of assignment mark total; 10% for the first two questions, 15% for the final question]

There are not necessarily right or wrong answers to the questions in this section, what is asked for here is evidence that some thought has been expended on the answers. You may use AI tools for research, but your answers must be written in your own words.

What is a Project?

For the following activities or organisations, state whether or not they fulfil the criteria given in lectures for being a project and provide a 1-2 sentence justification for your answer. You may need to do a little reading about some of the organisations in order to make an informed judgement.

1. Building a 3D model (e.g. a model aircraft, model railway, model building, etc.)
2. Doing the dishes
3. Installing a dishwasher
4. Open source projects (e.g. the MicroPython Project, Thunderbird Project, etc.)
5. The Alan Parsons Project
6. The Manhattan Project

What is Business as Usual?

Explain in your own words, in 2-3 sentences, what is meant by "business as usual" and how it differs from "a project".

Engineering Design

This year's project is a moderate *development design*, rather than being a *new design* (definitions of these terms are given in section 1.2.1 of *Engineering Design Process* by Yousef Haik and Tamer Shahin; free PDF available from https://www.researchgate.net/publication/235959233_Engineering_Design_Process). Both require technical expertise but a *development design* builds on an existing body of design work while a *new design* does not. There are advantages and disadvantages to a *development design* project compared

to a *new design* project. In your own words, in 2-4 sentences, what advantages and disadvantages do you see between a *development design* project and a *new design* project?

Source Control Management

[50% of assignment mark; 10% for each numbered question in the subsections below]

Answer the following questions assuming a context of a clean (no modified files) working copy. You will find most of the answers are in the online git documentation https://git-scm.com/docs, particularly the git glossary and the entries for specific git commands (e.g. `reset`, `switch`, `checkout`, etc.). You may use AI tools for research, but your answers must be written in your own words.

You are encouraged to experiment with your own working copies of the `data-recorder` repository on the ECS workstations. If you're worried about mucking-up the clone then it's perfectly acceptable, even encouraged, to make a separate "pristine" clone of the `data-recorder` repository for experimentation.

Working with the Commit History

Detached Heads

Executing `git checkout <commit>` on the command line produces a large amount of "advice", (106 words across 16 lines as of git 2.39.0), mentioning in an ominous tone a "detached HEAD" state.

1. What is the HEAD?
2. What is a "detached HEAD" state?
3. What *specifically* does the recommended command to recover from a "detached HEAD" state `git switch -` do, i.e. what does the `-` mean and how does it behave in practice?

Reverting the Working Tree to an Earlier Commit

On many occasions it will be helpful to revert the working tree the state in a previous commit, either temporarily or permanently. In addition to the online documentation, you may find the explanations in the Pro Git book helpful in answering these questions.

1. What is the difference between the three forms of `git reset`: `--soft`, `--mixed` and `--hard`?
2. What is the difference between `git checkout <commit>` and `git reset --hard <commit>`?

Challenge Questions

[15% of the assignment mark; 5% for each question below]

The questions in this section require a deeper understanding of how `git` functions as a source control management tool. Attempting them is worthwhile and recommended, but they are more challenging!

1. *Ancestry References* are relative commit references and there are (of course) two forms with subtly different behaviours: the `^` (caret) or `~` (tilde). What is the difference between the two? For the branching strategy used in the course methodology, are there any commits for which this difference will be important?
2. There are two ways (actually, more) to destructively remove the most recent set of local commits more recent that a specific `<commit>` reference: `git reset --hard <commit>` and `git rebase --interactive <commit>`. What are the differences between these two forms? Which form gives slightly greater protection when destructively removing commits?
3. The *RefLog* is useful when a mistake has been made and previous versions of files need to be recovered. What is the RefLog, i.e. what does `git reflog` show *and* what is the most basic limitation in its ability to be used to retrieve work deleted from the working copy?
   Hint: the working copy may contain files which are *untracked*, *tracked* and *tracked, modified but with changes uncommitted*: which of these does the RefLog record?
