Procedure for project 1

You proceed as follows. But, also refer to my class remarks. Remember that you are supposed to teach one another, learn from one another, collaborate, etc.

1. First, your team selects a member to **set up basic items to get the project going**.
   I call this selected team member **ABC** below.

2. On WSL, **ABC** fires up VS Code. Assuming that the **Kanpilot Kanban** extension has been installed already, **ABC** opens this extension. It will ask to initialize with a config file (it says: Config Not Found!; Initialization; so click the button).
   It creates a file named ***kanpilot.toml***, with some contents already. Do not write any additional contents onto it at this point.

   <u>Pay attention to my class remarks about Kanpilot Kanban</u>

3. Next, **ABC** creates an **empty file** named ***project_1_350.erd.json***.
   This file will contain the ERD design of your project.

4. Next, **ABC** gets into GitHub and creates a repository for the project, Name the repo as

   Project_1_350

   Add a README file. You should edit this README, now *or* later, describing your project.

5. Next, **ABC uploads** the above two files, ***kanpilot.toml*** and ***project_1_350.erd.json***, to that repo Project_1_350: on the project GitHub page, there is a "Add file" dropdown list, with an option to "**Upload files**".

After the upload, click the "**Commit changes**" button, to commit (on the only existing branch, ***main***, not a new branch under main).

6. Next, **ABC invites the other team members as collaborators**, as before.

The instructor must be a collaborator too:

(recall: my login name at GitHub is **manalym**).

7. As the project proceeds, there may be additional files to be uploaded and committed to the main branch (e.g. design files, SQL script files, etc.)

8. Now, to work on the project, at any time your team may use (or switch to) one of two modes:

A. Real-time team collaboration

Here, **ABC** alone clones the project repository to a local computer. **ABC** changes directory to the cloned local repo. Then, **ABC** makes a new Git branch (which will be the branch that the team will work on). Then, **ABC** starts VS Code from that directory (which becomes the workspace to be shared). **ABC** then opens the **Live Share** extension and proceeds as explained in previous classes.

The team then works on anything in the repo they want to work on.

When the collaborative real-time session is ended, **ABC** then

(i) stages and commits the changes (the work done)

(ii) pushes the branch onto the repo on GitHub (review **previous CLASSES**)

GitHub will say (a button) something like "Compare & pull request"). **ABC** may immediately do **a Merge** (of the work branch above and the main branch). You should find your way around here, but

refer to my class remarks as well

B. Asynchronous individual contribution

Here, a team member (I refer to as **XYZ**) clones the repo on GitHub, as usual. Then, as in assignment 1, **XYZ** creates a new branch and does work in that new branch.

When done, **XYZ** then does the usual: stage, commit, push to original GitHub repo. But, this time, **XYZ** does make a **PR (Pull Request)** to invite reviewers to review the work done in **XYZ** branch, before this latter branch is merge with main (both **ABC** and **XYZ** may perform the merge). I will not say more here, but

refer to my class remarks

There are additional commands that **XYZ** may elect to use (git fetch, git merge, git pull [pull = fetch+merge) that I will not explain at this time; read the **ProGit book**.

9. Now, the specific project for each team will be emailed to the team. Note, however, the following important items related to all pojects.

(a) Making visible "**all**" the work done for a project is very important in the management of the project. Tools like a Kanban board partly help with this visibility of work.

(b) All projects involve creating a **PostgreSQL** database.

(c) You start off by doing some research about your topic.

(d) You make an ERD model, using tools like ERD Editor

(e) You need to create many database schema objects (some I present in class, others I do not), for example functions and procedures, triggers, indexes, etc.

(f) You may not have all the real data needed to test your product. So, you may have to create some *fake* data. However, you must research sources of genuine data. And the fake data must be reasonable.

(g) You do not have data entry clerks in the project. But, **data entry**, although being a tedious and unappealing task, must be done anyway.

(h) But, there is no such a thing as a completely linear workflow in a project. It is **all iterative**. That is, you need to loop through **(c) – (g)** quite often.

(i) Of course, you cannot expect the instructor to do the project for you. In fact, the instructor may not have any answers at all.

<u>Again refer to my class remarks as well</u>