Summer Practicum

4 The Win

Best Practice Guidelines v.1.1

Introduction

In this document you will find a guide to the expected best practices for members of the research practicum team ‘4 The Win’, aka. ‘Team 4’.

This document has been broken into subsections related to the various areas that they are applicable to. In the event that a scenario is encountered that is not covered by this guideline team-members should use their best judgment to proceed in the spirit of the document and notify others as appropriate, until the document has been updated.

This Document

All updates to the document should be listed in this section. The message should say what sections where changed and have a brief couple of words on what was changed, mirrored in the Github message. For example if there is an update on how to manage the master branch it should be detailed in the Github section, but also listed below as version number and change, eg. v.1.1 Github Master Branch - branch management. v.0.# changes just need the message as sections are being initialised.

All changes should included an version incrementation as appropriate and all versions after and including v.1.0 should also be followed by the date in the form DD-MM-YY.

Small changes, such as addressing spelling and grammar, should be indicated with the addition of additional numbers after the second number in the version, eg. v.1.2.1. These increment until one of the other numbers reset at which point they are reset to zero which should be hidden. These changes do not been to be recorded below but a message addressing this should be included in the Github.

v.0.1	Document Initialisation	
v.0.2	‘Introduction’ and ‘This Document’ section added	
v.0.3	Heading for ‘Github’, ‘Coding Style’, ‘Team Decorum’ and ‘Conclusion’ added	
v.0.4	‘Github’ section expanded	
v.0.5	‘Passwords and Sensitive Data’ section added	
v.0.6	‘Code Respect’ updated	
v.0.7	This Document - Update of version recording practices	
v.0.8	Conclusion - Conclusion content added	
v.0.9	This Document - Protocol added for minor changes	
v.1.0	General - Document named and published to Github	21-06-21
v.1.1	‘Git Branch Management’ added	23-06-21

Github

Project synchronisation, version control, recording and backup should be managed using Github. The Github should comprise of a ‘Master’ branch which should only be updated with stable, and complete, features, and only with the permission of either the majority for the team or the maintenance lead.

At all times the working version should be the first child of master, the ‘Development’ branch. Once again team members should not work directly from the development branch, but should merge into it often when features they are working on are complete. At the start of a work cycle, or as needed, be sure to pull from this branch as it is the common branch and will be used to keep users in sync.

Individual working branches should be used, as childs of ‘Development’, not ‘Master’. When working on a feature a team member should branch off of development and merge back in when their feature is complete. Naming conventions for these at the discretion of each individual, and individuals may have as many of these ‘Feature’ branches as they can manage. A standard naming convention may be ‘feature/map’, ‘<name>/map’, or ‘feature/<name>’, but individuals can name them as they wish provided that it is clear what the mane means to external parties.

These feature branches should also be deleted when they have served their purpose to minimalism. These can be minutes or days later as the user needs, and branch names can be reused.

All Github updates should be accompanied by an appropriate message regardless of branch. Saying the file was updated is not helpful to others unless you say how it was updated.

A reminder, no work should be done directly onto the ‘Development’ or, particularly, the ‘Master’ branches.

Team members are also reminded to push their work, regardless of branch, to Github commonly. This creates a backup to reduce risk of work being lost, and reduces the risk of synchronisation issues.

Git Branch Management

In this section you will find a guide to basic branch management. All commands are presented here in italics and bold but I do not believe this will effect terminal commands. <name> is used to represent the name of your working branch. A general order of running these will be included at the bottom for easy copy/paste.

To check what branch you are on run ***git status***
This can also be done using ***git branch*** which will list all your local branches. Your current working branch will have a * next to it. ***git branch -r*** will also list remote branches.

To create a new branch use ***git checkout -b <name>***
This will create a new branch and move to it. There is a way to make a new branch without moving to it but mostly this is simpler as it save you having to then move into it. To move between branches use ***git checkout <name>***

Newly created local branches will need to be pushed to Github to be tracked remotely. To do this enter ***git push -u origin <name>***
Beyond this adding and committing files is unchanged. That said the push command is a little different as you need to specify where to push to. To do this use ***git push origin <name>***

When you are finished working on a branch and want to merge it into a parent branch checkout out the parent branch, the one you created the branch from.
DO NOT MERGE TO NON PARENT BRANCHES!!! I have no idea what will happen but cannot imagine it will be good if the system does not stop you. For example if you created the feature branch off of the development branch then checkout development.
The command to merge a branch into your working branch is ***git merge <name>***
This is where sync issues can happen. If so you will be asked to make decision on what code to keep and push again.

If the merge has been successful you can now delete the old branch. You will need to delete both the local and remote copies. To delete the local run ***git branch -d <name>***
In the event you want to delete a branch without merging it first this commend will fail. Instead run ***git branch -D <name>*** to force delete the local branch. These will only work if you are not in the branch.
To delete the remote branch run ***git push origin —delete <name>***

General commands listed in order:

```
git checkout -b <name>
git push -u origin <name>

git add *
git commit -m <message>
git push origin <name>

git checkout <parent name>
git merge <name>
git branch -d <name>
git push origin —delete <name>
```

Passwords and Sensitive Data

Passwords and sensitive data should never be stored in the Github. Instead they should be applied to files by importing them as variables from an external file, eg. SUS.py. This file should be named in .gitignore so that it is never synced to the GitHub.

Passwords and sensitive data should in general never be sent in a written or text format to others, such as through email or text.

Coding Style

Team members are free to code as they see fit. However a few styles are expected to be followed to prevent issues.

Variables should be clearly named. In the event that a variable is declared using an unclear name, such as an initialism, a comment should be left to explain the meaning of the chosen name. Commonly used variables, such as 'i' are acceptable without explanation. Multi-word variables are to be declared using camel case.

Comments should be allied commonly. They do not need to explain what a section of code does line by line, unless it is a complex piece of code that might be difficult for a user to decipher, but should explain what a block does. Similarly docstrings should be used for all functions to explain what the function does. If the function if long then comments should also be applied, but for small ones the docstring may suffice.

It is understood that external code, such as from the Google Maps API, cannot be commented completely, but a comment to say what the code does should be provided.

Code Respect

When it comes to editing, refactoring, or deleting code written by others check in with its original creators, or the team. There may be other code dependant on how this code works and, if nothing else, someone put a lot of work into the code and feel emotionally attached.

Conclusion

If anything is unclear or needs to be added or changed please be sure to discuss with the group so that this document, and team practices, can be updated.

Good hunting.