Scenario for using git, Github and the Github app

Brenda has a party shopping list which she creates as two simple text files (`drinks.txt` and `food.txt`) on her computer. She doesn't want to lose it if her laptop breaks/gets lost. She wants to be able to see the complete history of her changes to the files over time. She also wants to allow other people to see her party shopping list as it might be useful to them. git and github provide the perfect solution.

Brenda sets up her repository on her computer and on github:

Set up a repository

1. She puts `drinks.txt` and `food.txt` into a folder called `party-shopping-list` on her computer
2. She drags that folder to her github app and creates a new repository (this now exists only on her computer)
3. She clicks on 'changes' to see her current changes (the two files `drinks.txt` and `food.txt` have been added)
4. She creates a 'commit' (by writing a summary and clicking 'commit to master') to save her changes in git history in case she wants to come back to this revision of the files in the future
5. She publishes her repository to github (clicks 'publish') so that her commits are now stored on github so her changes are safe and other people can see her party shopping list

Her housemate, Barry, wants to make changes to the party shopping list. Barry also has a github account already.
Brenda allows Barry to make changes:

Add a collaborator

1. She goes to her github app
2. Clicks on the party shopping list repository on the left
3. Clicks on the settings icon

4. Clicks on ‘View on Github’ - she can now see the repository on the github website
5. She clicks on ‘settings’, then ‘collaborators’ and adds Barry as a collaborator

Barry now wants to edit the party shopping list on his computer:

Clone a repository from github to your computer

1. Barry opens the github app on his computer
2. He clicks the ‘+’ icon in the top left, then ‘clone’, finds the party shopping list repository and the clones it to a folder on his computer
3. He can now see the history of the party shopping list
4. He updates the party shopping list by opening the files (e.g. using sublime) and adding a couple of items to the food and drink lists before saving the files
5. He goes back to the github app and can see his changes in the ‘changes’ tab
6. He creates a ‘commit’ (by writing a summary and clicking ‘commit to master’)
7. He then clicks ‘sync’ to update github with his changes

Anytime Barry or Brenda now want to make changes they need to do the following:

Make changes and share them

1. Sync - to download any changes from github (from other collaborators)
2. Edit - make changes in their favourite text editor (e.g. Sublime)
3. Commit - save their set of changes on their own computer
4. Sync - upload the changes back to github so others can see them

Dealing with merge conflicts:

Step 4 above may not work if Brenda and Barry have both made changes to the same file or files at the same time. The person who syncs last must deal with a ‘merge conflict’.

Fix merge conflicts:

1. Open the files in a text editor (e.g. Sublime)
2. Merge conflicts will have lines around them that look like this <<<<<<<<<<
3. Choose the correct lines to keep
4. Edit the lines if necessary
5. Remove the lines with arrows <<<<<<<<<<<
6. Go back to the github app and commit the changes
7. Sync to upload the merged changes