Guideline in a nutshell

1) Overall idea
   a) How to handle requirement managing in a GitHub with a semi structured way
   b) Guideline mainly focuses on Issues –tool, but some thoughts about wiki [46-47] and testing practices [39] are also shared. [14]
   c) In order to make this guideline to work: everybody should know the basics of Issues –tool (those who manage requirements, must have a deeper understanding than others), programmers should know few special syntax tricks for commit messages.
2) Issues [14]
   a) Depict requirements and tasks. Both of those have two steps: main requirement (or just requirement) and a sub requirement + main task (or just task) and sub task [18-20]
   b) Hierarchical dependencies must be handled manually. A quick run through of valid scenarios: requirement can have sub requirements and/or main tasks, sub requirements can have main tasks, and main tasks can have sub tasks. Task can belong to multiple requirements but it must always follow the aforementioned structure (i.e. sub task must always belong to main task) [21-23]
      i) Minimum effort with hierarchy: child issues refer to parent issues. Preferred way: also parent issues refer to child issues. Issues can refer to others if there is a logical connection between them, which doesn't explicitly belong to the hierarchical structure. [23]
   c) Use labels and keep them up to date with every issue (concerns all stakeholders on their behalf). They are a critical part of the visibility and surveillance. [15-17]
      i) Checkout the general suggestion for label categories and individual labels. Remember category prefix for every label. [16]
      ii) Requirements should only have labels from type, status and misc categories. Priority can be assigned as a reminder but it is better to give it to the tasks themselves. [40]
   d) Utilize filters. Create bookmarks to the filter combinations you use the most. [33-36]
      i) Programmers: be aware of issues that mention you and those that are assigned to you. PMs should survey these things: issues currently being worked with, task lists, milestones, blocked issues, bug reports and enhancement proposals.
      ii) Milestone is a 'special filter' which has a deadline and issues can be assigned to it (issue can belong to only one milestone!)
   e) Update the description of an issue when needed! [37-38]
   f) Task lists should only be used in the description of an issue. [38]
   g) Programmers: always reference the issue your commit handles in the commit messages. Remember also to update the labels and task lists. [38]
   h) Thoughts on testing: should always be required. The same person, who implements a feature, should not test it. [39]
      i) Make sure everybody is aware of whom is authorized to close an issue
   i) For my recommendation of syntax for issues, see the open example issues in https://github.com/Ripppe/GraduRepo/issues?state=open (especially the task –type issues)
   j) Bug reports and enhancement proposals should be used when needed [41-45]
3) Issue creation step-by-step [25-27]
   a) Create title including possible reference number if such is used by the team (recommended)
   b) Write description

i) Check https://github.com/Ripppe/GraduRepo/issues/5 and https://github.com/Ripppe/GraduRepo/issues/6 **[26]!**

c) Assign labels
d) Assign people
e) Assign milestone (if needed)
f) Create the issue
g) If the issue was supposed to be referenced from another issue, go and update that issue now!

4) Where to start? [49-50]
   a) Go through this paper. Refer to the actual guideline when needed. Check also my example repo https://github.com/Ripppe/GraduRepo/ -> live examples with more explanation can be found there
   b) Decide what label categories to use, what are the colors for them, what are the labels themselves. Then create the labels.
   c) Create issue for every requirement and sub requirement. Remember the hierarchical structure. Follow the steps for creating an issue.
   d) Decide what kind of tasks are needed to complete each of these requirements
   e) Decide if those tasks should be split further to smaller pieces and create main tasks and sub tasks accordingly. Remember the hierarchical structure. Follow the steps for creating an issue.
   f) If milestones are needed, create them and assign issues
   g) Start using other principles of the guideline!