Guideline in a nutshell

1) Overall idea
   a) How to agile handle requirements managing in a GitHub with a semi structured way.
      i) Lean Software Development principles aid in agilism.
   b) Guideline mainly focuses on Issues –tool, but some thoughts about wiki and version control are also discussed.
   c) In order to make this guideline to work: users should know the basics of Issues –tool (those who manage requirements, must have a deeper understanding than others), GitHub and GitHub Flavored Markdown (GFM) syntax.
2) Issues
   a) Depict requirements and tasks. Both of those have two levels: main requirement (or just requirement) and a sub requirement + main task (or just task) and sub task.
      i) Whether to use sub tasks or main tasks coupled with task list depends on the situation and circumstances on hand: the bottom line is that enough relevant information is always there.
   b) Hierarchical dependencies between issues must be handled manually.
      i) 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. A task can belong to only one requirement and it must always follow the aforementioned structure (i.e. sub task must always belong to a main task).
      ii) Minimum effort with hierarchy: child issues refer to parent issues. Preferred way: also parent issues refer to child issues. Issues can refer to other issues that are not ancestors or descendants of the referencing issue and if there is a logical connection between them.
   c) Use labels and keep them up to date with every issue (concerns all team members on their behalf). They are a critical part of the visibility and monitoring.
      i) Checkout the suggestion for label categories and individual labels. Remember category prefix and color coding for every category.
      ii) Requirements should only have labels from type, status and miscellaneous categories. Priority can be assigned as a reminder but it is better to give it to the tasks themselves.
   d) Utilize filters. Create bookmarks to the filter combinations you use the most.
      i) Developers: be aware of issues that mention you and those that are assigned to you. Project managers should pay careful attention especially to the following: 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! *The description must be up-to-date.*
   f) Task lists should only be used in the description of an issue.

g) Developers: always reference the issue your commit handles in the commit messages. Remember also to update the labels and task lists.
h) For a recommendation of syntax for issues, see the open example issues in https://github.com/Ripppe/GraduRepo/issues?state=open (especially the task –type issues).
i) Bug reports and enhancement proposals should be used appropriately.

3) Issue creation step-by-step.
   a) Create title including possible reference prefix (recommended).
   b) Write description.
      i) Check https://github.com/Ripppe/GraduRepo/issues/5 and https://github.com/Ripppe/GraduRepo/issues/6
   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?
   a) Go through this paper. Refer to the actual guideline when needed. Check also the example repository https://github.com/Ripppe/GraduRepo/ -> live examples with explanations 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 first requirement and sub requirement issues to the issue tracker. Remember the hierarchical structure. Follow the steps for creating an issue.
   d) Decide what kinds 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 utilizing other principles of the guideline!