Don’ts
<h2>Search for shortcuts</h2>

There is no shortcut to GSoC selection and there is no shortcut to GSoC completion.

The student stipend is only paid after the contestants submit a minimum amount of code. Even if you do find a certain shortcut, you will not receive your stipend without presenting meaningful code.

If you are the right candidate for GSoC, you will not need a shortcut anyway, as you will love completing your task.
</section>

<section>

<h2>Disappear</h2>

If your mentor thinks you have disappeared during the summer, this tends to quickly result in failure. The most common problem is failing to mention long family vacations or class schedules in the summer. Disappearing includes taking the initial payment and running with it; if you're tempted to do this, you might want to consider the damage to your reputation, or the excited students missing out on a slot so you can waste yours.
</section>

<section>
<h2>Take another major commitment</h2>
The GsoC is a full-time job, and anything that takes more than 5 hours a week will cause problems. Having two simultaneous full-time jobs is unrealistic and is not expected of you.
</section>
<section>
<h2>Submit code at the last minute</h2>
Showing your code in public can be scary. Some students wait until the very last minute to show their code to their mentor and other contributors to the project. Do this only if you have a burning desire to fail, because it will be too late for any review to fix holes in your code.
</section>
<section>
<h2>Being shy about asking for help</h2>
This is a learning experience, and there is no such thing as a stupid question. You will face issues and problems that need to be solved, but you will need help. The mentors are here to provide feedback as well as help you when you're facing an issue, so never hesitate.
</section>
<section>
<h2>Consider this is a solo project</h2>

This is not college or high school, Google summer of Code heavily implies working with a community. The mentor will provide feedback and help you with your issues, while the rest of the community will keep contributing code, making the software a living entity that grows and evolves.
</section>
<section>
<h2>Commit multiple non-related changes at once</h2>
If something is bad about one of the changes and someone needs to roll it back, it's more difficult to do so. Make each commit succinct and functional, even if it means a little extra work.
</section>
<section>
<h2>Go for more than a week without communicating with your mentor</h2>
The project timeline doesn't allow for unplanned gaps in communication. Students should talk to their mentor at least once a week to update them on their status whether the mentor asks for it or not. That way, both of them can plan ahead and keep a steady work flow.
</section>
<section>
<h2>Start coding before planning</h2>
You can encounter numerous issues if you decide to start coding before discussing the project on an architectural level with your mentor. Changes in the plan will most likely translate into rewriting a bunch of code for no reason.

Do’s

<h2>Understand Free Software</h2>

You will have to interact with the community, write & submit code for the duration of the summer, so you are expected to have an understanding of what Free Software is. Click here to learn more about its philosophy.

</section>

<section>

<h2>Consider GsoC a full-time job</h2>

Unless otherwise discussed, students are expected to work about 40 hours a week on their project. This is essentially a full-time job. If students can't work this much or if there are periods when a student will be away on vacation, then that needs to be discussed beforehand with the mentors or project administrator. The mentors shouldn't have to hunt you down to keep tabs on your activities either, it is your job to provide periodical reports on your progress; most recommended is a weekly progress report.

</section>

<section>

<h2>Learn to manage your time</h2>

Time management is a great skill to have, and a crucial one during GsoC. You are expected to work this as a full-time job, 40 hours a week, for the duration of the summer. This will, most likely, be your first full-time job and working from home might cause distractions. The most important thing is to separate the "work" environment from the "home" environment, as to avoid distractions. Create a timetable with goals, milestones and have a daily schedule to keep you on track and optimize your time worked, while balancing it with your personal life.
The student should schedule time to work on the project each day and keep to a regular schedule. It's not acceptable to fiddle around for days on end and then pull an all-nighter just before deadlines. It will show in the code and in the evaluation.
</section>

<section>

<h2>Commit early, and commit often</h2>

This allows issues to be caught quickly and prevents the dreaded one-massive-commit-before-deadline problem. Developers can rarely ever commit too frequently. They can very easily commit too infrequently. Once a day is far too infrequent.
</section>

<section>

<h2>Commit with meaningful messages</h2>

The history in version control is frequently the best timeline log of what happened, why, and who did it. The commit messages should be detailed and informative, with references to specific bug numbers, links and issues.
</section>

<section>

<h2>Communicate with your mentor</h2>

The mentor is one of the most valuable resources to students. The mentors are generally already solid contributors with a long track record of involvement with the project. The mentor likely has worked on the project for long enough to know the history of decisions, how things are architected, the other people involved, the process for doing things, and all other cultural lore

that will help the student be most successful. Frequent communication with your mentor is a must. The student should make sure the mentor has a good idea of:

What you're currently working on
How far you've got
How you are implementing it
What you plan on working next
What issues have come up
What you did to solve them
If you're stuck, what the problem is
</section>

<section>

<h2>Let your mentor know what your schedule is</h2>

As previously stated, long periods of time with no communication often lead to failure. Make sure your mentor knows when you are going on vacation or trips that might stop you from working for extended periods of time, so that he can plan accordingly.
</section>

<section>
<h2>Be self-motivated</h2>
The student is expected to be self-motivated. The mentor may push the student to excel but if the student is not self-motivated to work, then the student probably won't get much out of participating. The mentors are there to help, but they're not supposed to be a crutch or substitute for research and hard work.
</section>
<section>
<h2>Plan ahead of coding</h2>
Work with your mentor to define the architecture of your project before you begin coding. You don’t need to go as far as prototyping every function, but you should have a vision of how it will all eventually work at a reasonable level of detail, such as important data structures and algorithms. Determine the libraries and tools you’ll use, and be able to justify your choices.