

**Mentorship Program**

**Mentor Guide**

**Linux Foundation**

**Open Mainframe Project**



Contents

[Role of the Mentor 3](#_Toc25786186)

[What Makes a Good Mentor? 3](#_Toc25786187)

[What to Expect from Under-Mentoring 4](#_Toc25786188)

[Mentorship Program At-A-Glance 4](#_Toc25786189)

[Identifying a project 4](#_Toc25786190)

[Mentee Selection Process 5](#_Toc25786191)

[Mentee/Mentor Bonding Period 5](#_Toc25786192)

[Goals of the kick-off period: 5](#_Toc25786193)

[Setting Expectations 7](#_Toc25786194)

[Defining Success 7](#_Toc25786195)

[Managing Output 8](#_Toc25786196)

[Workflow 8](#_Toc25786197)

[Creating a Strategic Plan 9](#_Toc25786198)

[Menteeship day-to-day guidelines 10](#_Toc25786199)

[Following the Strategic Plan 10](#_Toc25786200)

[Delivering Feedback 11](#_Toc25786201)

[Managing week-to-week progress 11](#_Toc25786202)

[Midterm Review 12](#_Toc25786203)

[Final week 12](#_Toc25786204)

[Final Review 13](#_Toc25786205)

[Resources for Mentees/Mentors 13](#_Toc25786206)

[GitHub 13](#_Toc25786207)

[Slack 13](#_Toc25786208)

[Cloud Instance 14](#_Toc25786209)

[Further Resources 14](#_Toc25786210)

Thank you for your willingness to participate as a mentor in the Open Mainframe Project Mentorship Program. This document is a guide help you through your role as a mentor.

Much of the inspiration and content for this manual comes from the [Google Summer of Code Manual](https://developers.google.com/open-source/gsoc/resources/manual), a big thank you to Google for providing this valuable resource.

# Role of the Mentor

Mentors are the key to mentee success. Mentees (the interns) will learn from you, not only the technical skills needed to complete their projects, but also the communication, planning, and scoping skills. These things will give your mentee(s) confidence. Remember, most of the mentees have never completed a project outside the classroom before, so while they are technical strong they maybe be looking for your wisdom and expertise to help guide them through their project during the mentorship period.

## What Makes a Good Mentor?

Mentoring a mentee can be a very rewarding experience. However, being a good mentor is not just a matter of winding up the mentee and watching them go. Quality mentoring requires a substantial time commitment and a willingness and ability to take a leadership role.

There are specific skills that you can work on in order to be more effective; even experienced mentors can improve.

* **Work to understand the technology and aspects of the developer community.** Mentees are coming new to this and look to you as a mentor to help them both understand the focus area they are working on and help build their own skills in learning how to navigate technology spaces and developer communities. If the technology area or community is new to you, you should look to build your own support network in the respective space. Having a stable of Subject Matter Experts to draw on is always helpful. No one expects you to be proficient everything.
* **Interest in the project's success.** If this project area isn't interesting to you, or if the results of the project are something you don't see value in, you are less likely to be excited to support the mentee and help drive them to success.
* **Dedicate significant time to the mentee.** Each mentee will have varying degrees of experiences and skills, regardless, you should be prepared to spend a significant amount of time helping the mentee. In a good mentee and mentor relationship, the time commitment will likely reduce as the project progresses, but it's better to be prepared to devote a significant amount of time throughout the project
* **Interest in mentoring mentees.** The overall goal of the program is for mentees to be mentored. Mentoring is important to cultivating the future of open source software. We do this through mentoring our immediate projects and the overall Open Source culture. Remember, you are training the next generation of open source developers. Mentoring a mentee requires a combination of passion, responsibility and patience. A good mentor is willing to engage with mentees throughout their learning process.

## What to Expect from Under-Mentoring

* **A poorly defined project.** The project goals and deliverables are unclear, and the work schedule is not set. The consequences of this are serious and impact the project if left unchecked.
* **Lose track of what the mentee is doing.** The state of the project is unclear, and its progress is uncertain. Evaluation is impossible to do well. The mentee proceeds down the “wrong path” and wastes valuable time, gets frustrated and discouraged.
* **The mentee produces code that isn't useful.** The mentee starts off on 'the wrong path', failing to use existing functions or established project idioms. Rather than fixing problems as they arise they keep on adding more. The code is never integrated into the main codebase because it doesn't work well enough and would require more work to fix than it is worth.
* **The mentee gets stuck.** The mentee seems to be engaged, and to be working hard, but no apparent progress is being made. Alternatively, the mentee's communications are infrequent and terse, and seem to always be on the same issue or milestone. Plan on spending some quality time going over details and code reviews.
* **The mentee disappears, perhaps for days or weeks at a time.** If the mentee is under-mentored, it may be difficult to determine when this period began, and thus to know when to panic. Insufficient information is available for evaluation; thus it becomes impossible to fairly evaluate the mentee. The mentee must be reminded that they have made a commitment to the mentorship, just as you have as the mentor.

# Mentorship Program At-A-Glance

1. Identifying a project
2. Mentee Selection Process
3. Mentee/Mentor Bonding Period – Project Kick-off
4. Midterm Evaluation
5. Final Evaluation
6. Project Wrap-up

## Identifying a project

Before the OMP mentorship is opened for mentees to apply, the OMP mentorship will solicit OMP members to submit project proposals. (The OMP is also open to non-members proposing projects.) Anyone can submit a project proposal, however, OMP projects (e.g Ambitus, Atom, ADE, Feilong, Zorrow, Zowe, etc.) and by OMP project members will be given priority. To submit a project proposal the project sponsor should fill out the OMP Mentorship Project Proposal form or submit a one-page description of the project.

Once project proposals are identified, a mentor is identified for each of the project proposals, many times the mentor will be the OMP member, or sponsor, that proposed the project or someone from their organization. After mentors are chosen and project feasibility has been determined, the project will be posted and offered to mentee applicants as a project that they may choose as a project they are interested in during the OMP mentorship’s mentee recruitment phase.

## Mentee Selection Process

As a mentor, you have expertise in one (or more) of the area(s) of the project you’ve been selected. The Linux Foundation’s Open Mainframe project will post your project(s) in the mentee application form for the summer mentorship applicants. Applicants can choose from a number of projects offered for the summer mentorship. Based upon some basic criteria, a set number of applications will be chosen for your project’s final consideration. The mentor of a project may select the mentee applicants they’d like to put on their short list for their project.

The mentor reviews the list of applicants on his short list for his project and conducts individual interviews with the mentee applicants. The purpose of the interviews is to determine the mentee’s qualifications for working on the projects. Such things as skills, aptitude, commitment, etc. should be reviewed by the mentor. The mentor then ranks the applicants in the order of preference. A mentor may select one or more mentees to work on his/her project based upon the number of mentees needed and/or the number of mentorships available for the current summer’s mentorship program.

## Mentee/Mentor Bonding Period

The mentorship program starts with a two-week kick-off or bonding period, allowing the mentors and mentees a chance to learn how to work together.

### Goals of the kick-off period:

* Prepare mentees by introducing them to the resources they will be using on the project.
  + Establish project milestones and expectations.
  + Identify the Linux Foundation Github repository, how to use Github as a code repository.
  + Identify the Github directory structure:
    - Which directory to place documentation
    - Which directory to place source code
    - Which directory to place notes/research
    - Which directory to place status reports
    - How to upload code and comment code
  + Set up a Slack channel for communication
  + Identify a cloud instance where they can code, implement, and test
    - Ensure that mentees have a development environment set up.
    - Ensure mentees get set up with the project version control system
    - Ensure mentees have access to the necessary documentation.
* Set up a working relationship. Get the mentees engaged socially in the project.
  + Identify a time each week to meet, via conference call or video conference
  + Video conference is preferable so that the mentor and mentee can share documents and code in real-time.
  + Be sure to put the meeting on your calendars so you don’t forget
  + Be sure mentees understand their commitment to the project.
* Work with the mentee so they can become familiar with the development environment. Guide mentees through the environment, don’t assume they will be able to do it themselves without guidance.
* Refine the strategic plan for project completion and submit it with mentor approval to the Open Mainframe Project Mentorship Program Administrator. Set up high-level milestones. This should be the beginning of the project design document.
  + Define the problem to be solved
  + Define the current state
  + Identify high level requirements
  + Define high level timeline
* Gather and fill out the required forms:
  + Tax forms required by the Linux Foundation
  + any contributor license agreements
  + any paperwork that your project requires
  + Signed Mentee agreements

Successful completion of your mentee's project depends a lot on the kick-off bonding period. Make sure that you and your mentee make good use of this time. Make significant progress on preliminary tasks, setting up expectations and the scope of the project. The mentee/mentor bonding period is also a good chance for the mentees to start interacting with each other. Early connections can help the mentees support each other during design and coding.

Ideally mentees are ready to start the design and code writing at the official start of coding, and are already engaged socially. During the mentee/mentor bonding period mentees are expected to learn about the project they are working on and of the Open Mainframe Project, ensure they have a development environment setup, get set up with GitHub (with the appropriate folders) and Slack, read up on necessary documentation, and further refine the strategic plan and design for successful project completion.

Plan weekly activities for your mentee that only take an hour or two. The community bonding period occurs when mentees are likely to still be taking classes and have a full load of coursework. It's unreasonable to expect that the mentee will be available full-time to work on the project at this point. Give the mentee assignments to read specific documentation or do particular activities.

If or any reason, the mentor believes that the mentee candidate is not a good fit for the mentorship, the Open Mainframe Mentorship Coordinator should be contacted. Otherwise, the mentee’s will receive their 1st stipend payment at the end of the kick-off period. Mentees should be engaged and ready to start of coding after they receive the initial payment. If you don't hear anything at all from your accepted mentee during the mentee/mentor bonding period, or the mentee explicitly drops out, let us know immediately.

### Setting Expectations

Successful mentors set expectations at the start of their projects. This includes communication frequency, project goals, availability and ways of delivering feedback. While the mentor should take the lead in expectation setting, the process of creating and documenting the expectations must be collaborative. Mentees and mentors need to agree on what is expected, or success becomes quite difficult.

### Defining Success

Performance measures make it easier to provide feedback to help your mentee stay and/or get back on track if s/he veers off-course. Clearly stated performance measures you make a fair determination on whether or not a mentee needs to be corrected or even removed from the program.

* **Get mentee input**: Make sure your mentee has input into the types of performance measures used to determine success or failure. It is very important that your mentee helps create the performance measures to determine project success and failures. Your relationship should be a highly collaborative one.
* **Set achievable goals**: Help your mentee come up with manageable project goals. Rather than defining the project as one giant chunk, help your mentee break the project goals down into smaller pieces or "inchstones" that allow a change in direction if necessary. It is sad to work the entire summer on one giant deliverable, only find out in the last few weeks that the architecture or design is defective. Have an overall plan, but use an agile development process.
* **Anticipate time away**: Make sure to set expectations for known or planned time away from the project, such as coursework, vacation trips or holiday time. Talk about how many hours or deliverables per week would be reachable goals and what amount would be a good stretch goal.

### Managing Output

Decide in advance what happens when project goals aren't met. Remember to be flexible if your mentee has made good progress or has obviously worked hard but needs to re-scope the project at mid-term. Good project management is hard. Your performance measures will help you manage project modifications.

* **Plan for Slippage**: Have a plan to deal with scope-creep and timeline slippage. What if something happens that prevents your mentee from working successfully for an extended period of time? At which point do you need to terminate the project? Have a plan in place for these scenarios.
* **Gather Feedback**: Your mentee's wishes and desires for a successful project are as important as the project goals. Make sure that you solicit and incorporate his/her feedback when coming up with initial goals, performance measures and communication methods.

Overall, communicate and be reasonable when it comes to your mentees. Be ready to revise project plans if an unexpected requirement or bug occurs.

**Pro Tip:** Ask about the weather and local stability of public services. Is your mentee using the cafe down the street for access to the internet for accessing his code/Github/documentation? Are there seasonal weather conditions that may lead to flooding and the subsequent inability to turn-on one's computer? Work on a plan to address these types of environmental issues that can affect both communication and output.

**Don't Be That Person:** No one likes dictators. Work with your mentee on the development of expectations, rather than barking out orders. Be a guide. The program is a learning experience for the mentee.

### Workflow

Make sure that your mentee is familiar with the workflow for the project as early as possible. Ensuring the mentee is familiar and able to work with the tools associated with the project, such as libraries, along with GitHub and Slack. Discussions with your mentee should also encompass code review, talking with other developers about which algorithm is best for the problem at hand and other meta-questions about how to best get from specification to implemented solution.

The best kind of workflows for the mentee projects are iterative. That means that small, quantifiable goals are defined and then acted upon. For instance, an example of an iterative workflow is:

* Write a test that demonstrates what feature will be added.
* Run the test to verify that it fails in the way you think it should.
* If it fails in an unexpected way, your test may be wrong. This is a great time to ask your mentor for guidance.
* If it passes, you are done! You just added test coverage to an already existing feature, and that is great!
* Add the feature (also known as "a simple matter of programming").
* Run the test to verify that it passes.
* Write documentation about your feature.
* Rejoice appropriately.

This iterative workflow is known as Test Driven Development (TDD). This workflow gets applied to each feature that will be implemented, so a TDD workflow will consist of many cycles of the above steps, each for a different feature. The polar opposite of this workflow looks something like this:

* Write design the first 1/3rd of the summer
* Write code and test the next 1/3rd of the summer
* Write more tests and document for the last 1/3rd of the summer

This is almost always doomed to failure, since people are always optimistic about time estimates for completing something. Sometimes you must debug a weird issue. You can't predict how long it might take to resolve the issue. What usually happens in this workflow is that the code gets written, but takes longer, half the tests get written and no documentation is written because the mentees have run out of time.

Take an approach which produces usable code even if parts of your plan fail, or are never attempted. Be humble and flexible about your development model. The mentee may teach you something!

**Pro Tip:** Progressive milestones may allow code to be merged progressively.

### Creating a Strategic Plan

To increase the probability of project’s success you should spend time with your mentee refining the plan developed in the application period. This should be done during the mentee/mentor bonding period. A good strategic plan includes:

* **A high-level design document**: Think about the architecture and design of the new feature or enhancements that you are making to your community's project. Take some time to teach your mentee to think about usability by writing a few quick use cases and scenarios, and consider the introduction of any new dependencies.
* **Progressive milestones that build on the previous work:** At the completion of each milestone you and your mentee should take the opportunity to celebrate the accomplishment and reflect upon it. Using progressive milestones also gives you a good measure of how far you have come, which can be very useful during periods of frustration. Additionally, if you don't reach all of the final goals of the project, you will have tangible achievements to point to when reviewing progress with your mentee. Reinforcement of a mentee's tangible accomplishments encourages them to stick around and helps to create life-time contributors.
* **Target completion dates for each milestone**: In reality, completion dates are going to move. Nonetheless, a target date gives you a time frame for closure and helps control "scope creep". Coach your mentees in how to recognize that a milestone is going to be missed, and to notify the project participants before the dates passes.
* **Tasks associated with each milestone**: Because your milestones are most likely going to be chunks of of code, each milestone needs to include both testing and documentation around that particular "chunk". This approach helps guarantee that you and your mentee don't end up with a pile of code that hasn't been tested or documented at the end of Open Mainframe Project Mentorship Program.

Leveraging the GitHub wiki functionality is the recommended place for capturing this plan and ensuring everyone has visibility into it.

## Menteeship day-to-day guidelines

After the date on which the mentorship program commences, mentees are expected to work 40 hours per week on the project (unless this is done as a class project - in that case consult with the academic institution instructor for guidance.)

### Following the Strategic Plan

Since you created a strategic plan during the mentee/mentor bonding phase, you actually need to follow it. Some ways you can use your strategic plan to stay on track are:

* **Collect regular status reports**: Status reports are an important communication tool. They are also important in making sure that time-line slippages and scope creep are addressed in a proactive manner. You do not want to find out two weeks after a milestone due date that your mentee has slipped the date because they have been unable to solve a simple problem. Status reports should be done every week (or minimally every 2 weeks.) Status reports do not have to be long – just (1) what the mentee accomplished the last week, (2) what the mentee plans to accomplish the next week, and (3) what difficulties or issues the mentee is having. It is highly encouraged that the mentee document the status plan and file it in the GitHub directory.
* **Check off associated milestone tasks**: Find a way to keep track of tasks, and then indicate when they are completed. This can be as simple as keeping an informal to-do list in that is referenced in weekly status reports. You can also use the project management software that the rest of your organization uses. Do what works best within your community, but make sure you do something.
* **Set an expectation of prior notification of missed deadlines**: Your mentee is going to miss project deadlines. Make sure that they understand that it is important to notify you of the missed deadline well in advance. Understanding how long something is going to take to complete is a valuable skill, but it is one that is learned through ongoing evaluation.
* **If your mentee misses a deadline, make sure you discuss why**: Helping your mentee understand where they become bogged down or stuck helps with future strategic plan development. Remember that you are cultivating a long-time contributor.
* **Be a good example**: If you, the mentor, need to miss a deadline, make sure you communicate this to your mentee well in advance.

### Delivering Feedback

Throughout the project you should be delivering effective feedback to your mentee about their code, communication, and documentation.

* **Deliver feedback early**: Don't wait until several issues have come up, or until your mentee has impressed you multiple times with their efficiency. Let them know right away what you think about their work.
* **Make a point to give positive feedback:** The open source community parcels out compliments all too frugally. When a mentee completes a task on time, and especially when they exceed your expectations, let them know! Early praise is a far better motivator than late criticism.
* **Don't avoid critique, but don't be a jerk:** Try to put yourself in your mentee's shoes, and consider how you might want to hear constructive criticism. Phrase suggestions positively. If criticism is personal in nature (i.e. tone of an email, timeliness or other non-code issues), deliver it in private rather than in a public forum. When in doubt, ask for advice from more experienced mentors or from your organization's administrator.

**Pro Tip:** Don't let the development of your design document become your Open Mainframe Project Mentorship Program project. You should have the strategic plan completed by the end of the mentee/mentor bonding period (or by the end of midterm) so that you can start coding right afterwards.

**Don't Be That Person:** Don't be overly-critical of date slippage. It happens. Fanatical adherence to dates does not lead to successful project completion, nor does it make your mentee feel excited to contribute to your community long-term.

### Managing week-to-week progress

Just like using the GitHub wiki for doing the project strategic plan, a great practice is to document progress weekly in the GitHub wiki. This doesn't need to be a complex in such a way that creates a burden on the mentee or the mentor; a simple wiki page with a list of milestones completed, milestones to be addresses in the next week, and milestones yet to do is perfect and helps structure the weekly meetings. Here's an example:

Done in the past week

- Added function X

- Fixed outstanding issue Y

To do this week

- Add new feature A

- Fix regression B

Still outstanding to do

- New Feature C

- New Feature D

## Midterm Review

Towards the middle of the mentorship period, the mentor and mentee should come together and discuss the overall progress of the project and growth of the mentee. To facilitate this, there will be a standard system created to capture this feedback, encompassing of

* Mentee evaluations: Each mentee fills out a mid-term evaluation pertaining to their experience and their interaction with mentors.
* Mentor evaluations: Each mentor fills out a mid-term evaluation covering their participation and their mentee's performance.
* Mentee is expected to have a detailed design by mid-term evaluation – a successfully verified detailed design will signal the issuing of the mid-term stipend.

Output from this mid-term review will determine both (a) whether the mentee continues in the program and (b) mid-term payment to the mentee.

There is great guidance and advise for mentor reviews at <http://write.flossmanuals.net/gsoc-mentoring/evaluations/>.

### Final week

The final week should be spent cleaning up all loose ends on the project, such as:

* Documentation
* Test suites/cases
* Presentation on work done

The Open Mainframe Project will coordinate working with mentees with their presentation development and strategy. As a mentor you should help your mentee put together their presentation and prepare them to speak on the project.

## Final Review

Once again at the end of the project both the mentor and mentee will come together after each completes their respective evaluation.

* Mentee evaluations: Each mentee fills out a final evaluation pertaining to their experience and their interaction with mentors.
* Mentor evaluations: Each mentor fills out a final evaluation covering their participation and their mentee's performance.
* Mentee presents his final coded project to mentor. After verification of a successfully completed project and the scheduling of the conference presentation, the final stipend will be issued to the mentee.

Much like the mid-term review, output from the final review will determine both (a) whether the mentee completed the program successfully and (b) final payment or credit to the mentee.

# Resources for Mentees/Mentors

The Open Mainframe Project provides the following tools for both project management and collaboration.

## GitHub

The [Open Mainframe Project Mentorship Program GitHub organization](https://github.com/openmainframeproject-internship) will host all mentee projects. GitHub provides source code management, issue tracking, and wiki collaboration tools that should be leveraged by the mentee and mentor during the program.

**Action for Mentor:** When the project is determined, ensure the mentee works with the Open Mainframe Project Mentorship Program Administrator to create the repository for the work ( either using an empty repository for the case of new projects, or a fork of the upstream project for work on existing projects ). You should provide your GitHub ID to the Open Mainframe Project Mentorship Program Administrator as well so you can have the ability to review code easily.

## Slack

The Open Mainframe Project has a [Slack service](https://slack.openmainframeproject.org) for community communication, and encourages all mentee/mentor communications to leverage Slack. Slack provide instant messaging, audio/video calling, and recording of historical conversations.

**Action for mentee and mentor:** Sign up for a [Slack account](https://slack.openmainframeproject.org). The Open Mainframe Project Mentorship Program Administrator will create a channel for communications specific to your project.

## Cloud Instance

Students can leverage the [LinuxONE Community Cloud](https://www.openmainframeproject.org/trymainframe) for a development and testing environment for their projects. These instances are available at no charge and allow a free instance for use for up to 120 days.

**Action for mentor**: Sign up for [an account](https://www.openmainframeproject.org/trymainframe) or help provide your mentee access to a mainframe system for development.

## Further Resources

More details on mentor best practices and process are found in the [Google Summer of Code Mentor Manual](http://write.flossmanuals.net/gsoc-mentoring/about-this-manual/). This program was inspired by the GSoC program.