# Step by Step Guide for Requirements/Design Tasks

The purpose of this guide is to describe in a general way what the teams should do during the requirements and design tasks. The steps described below are general suggestions only. Your team might find that the steps need to be tailored/modified in order to adequately address the needs of the tasks.

1. **Read the memo:** You should first try to understand the nature of these tasks. This is going to give you the high level context that you need in order to develop the plan. You have a short time to accomplish the task and need to focus your activities. If you start exploring all of the resources prior to gaining an understanding of what you need to do, you’ll spend time unnecessarily on activities that don’t directly support your effort.
2. **Establish a development process:** You want to avoid proceeding in an ad-hoc manner. You want to try to adopt a disciplined approach to understanding, designing, and building this system. We recommend an agile approach that will allow you to incrementally design, build, and validate your system. Something like SCRUM with weekly sprints seems like a good choice. If you don’t have any experience with SCRUM, you might want to execute a short initial sprint in order to validate your understanding of the process. The instructors for both task 5 and 6, or the program faculty can provide assistance if you are having trouble implementing SCRUM.
3. **Prepare requirement documents templates:** Requirement documents are formal and precise documents designed to be durable across individuals, teams, and time. Understand the structure of the deliverables, prepare templates, and grow them into the deliverables as you elicit the requirements through an iterative process of listening, documenting, reviewing, revising, reflecting, and refining.
4. **Elicit an initial set of requirements:** While some of the functional requirements seem pretty clear for this project, there are still many ambiguities. You’ll want to clarify any open questions about these requirements through iterative refinement. You’ll also need to elicit environmental and quality requirements. You’ll be given a list of the stakeholders that you’ll work with in order to understand the needs of the organization. Keep in mind that not all stakeholders have the same understanding of what’s needed. Facilitating communication toward resolving conflicts among stakeholders is a natural part of most development efforts.
5. **Create the overall architecture:** In order to realize the benefit of multiple people working on the project you’ll need to establish some means of coordinating. This means you’ll need to divide the work somehow, but be able to ensure that these independent work items are design and constructed in a way that they will fit together to realize a system that meets the intent. A high level architecture should not only enable the team to decompose the tasks, but also ensure that the resulting system will support the desired systemic properties.
6. **Identify inter team interfaces:** Once the major units of work have been identified, the specific responsibilities of each and the dependencies amongst them need to be worked out. Defining the interfaces of these units of work is a good way to ensure that everyone has a common understanding of the responsibilities of each unit of work as well as having a common means for accessing services provided by each element.
7. **Detailed design:** Once the overall structure of the system has been established and allocated to one or more team members, the internal structure of these elements needs to be worked out. As long as the external interfaces are adhered to, these decisions can be made locally without input from the rest of the team. While in theory once you define the interfaces, they don’t change, in practice, there are almost always occasions when the original interface definition is found to be lacking and needs to be updated.
8. **Implementation:** As the detailed design becomes sufficiently developed, implementation can begin as both it and the design are finalized in parallel. In the end, both the design and the implementation need to be completed and validated. Iteration is critical to success. The implementation should be unit tested throughout implementation just as the design should be validated as it develops.
9. **Integration and test:** Prior to the end of each sprint the system should be integrated and tested against the requirements. There are many ways to do this (e.g. continuous integration vs integration at the end of the sprint), but it’s important to do this regularly. Otherwise you run the risk of accumulating technical debt that can come back to haunt you later on.
10. **Reflect:** At the end of the sprint, you should analyze what worked well and what didn’t. You might find that you want to adjust your approach in order to avoid experiencing issues repeatedly.
11. **Repeat:** One of the primary risk reduction mechanisms in agile development practices is short iterations. These allow you to get feedback on a regular basis in order to adjust as need be. For a project of this duration it’s recommended that you have iterations of no longer than one week.