# Dabbler Guide 1: Project Planning

## Planning Software Projects the Dabble Lab Way

# Table of Contents

# Chapter 1: Introduction

We're always looking for ways to design, build, and ship software faster—without sacrificing quality or increasing costs. This goal isn't unique to Dabble Lab, but it's an elusive one. The promise of going further, faster, with less risk, is one of the primary reasons our clients hire us.

This Dabbler Guide is about effective project planning. For Dabblers, the processes outlined in the following sections are critical to success at Dabble Lab and beyond. For our clients, this guide provides context and clarity to how we approach projects.

Project planning—which could more accurately be called "expectations management"—is the process, and perhaps art, of clarifying and communicating stakeholder expectations. A project can only be successful if "success" is defined objectively before development starts. While too often overlooked, effective project planning is the single most important factor to going further, faster, with less risk.

At Dabble Lab, our approach to project planning is formulaic and thorough. We take sole responsibility to define and align realistic expectations for our client projects, so it's exceedingly important for Dabblers to follow our project planning practices avoiding costly mistakes.

Our project planning process has three general parts—each of which is given its own chapter in this guide:

1. Project Discovery
2. Requirements
3. Budgets and Timelines

Every project begins with expectations. As a project maintainer, your primary responsibility is clarifying stakeholders' expectations with the development team. Project discovery is an on-going process as client expectations will evolve and become clearer with time.

At Dabble Lab, project maintainers are responsible for documenting stakeholder expectations as requirements. Project discovery is on-going, but after an initial project discovery process, there should be general and functional requirements that can be objectively completed.

And finally, instead of estimating and budgeting for total projects, we estimate and budget each requirement.

After reading this Guide, you should be able to:

- Clarify project expectations (costs, timelines, and deliverables)
- Write objective requirements
- Course-correct from bad estimates

# Chapter 2: Project Discovery

Simply put, software projects fail because somebody's expectations weren't met. Whether those expectations were unknown, unclear, or unrealistic doesn't matter. At best, resources get wasted, and stakeholder confidence is eroded. At worst, trust is lost, and the project is canceled.

Projects should never fail because of unknown, unclear, or unrealistic expectations. Not all software hits the mark from a business perspective, but realistic expectations regarding business risks should be set before the project begins. Project discovery is about uncovering and detailing stakeholder expectations and is the first step in defining project success.

## Understanding Expectations

Even a small software project will have all kinds of stakeholder expectations. There will be expectations about functionality, usability, aesthetics, performance, scalability, maintenance, support, security, timeframe, cost, and many others. It's safe to assume that you won't uncover every expectation up-front. The project discovery process is ongoing, but you should always be pushing for more clarity. The sooner you uncover and clarify expectations, the better off the project is.

The most challenging part of the discovery process is that stakeholders often "don't know what they don't know." For example, you can't expect someone who isn't familiar with software security to have clear expectations regarding what security measures should be implemented. However, if the system gets hacked, you can reasonably expect them to feel like their expectations weren't met. As a project planner, it's unrealistic for us to assume a client will be clear about all of their expectations.

In addition to uncovering expectations, it's also our responsibility to understand what expectations stakeholders are clear on. Expectations can only be managed if everyone is on the same page.

## Expectation Types

Most stakeholder expectations fall into one of the following categories:

1. Deliverables: What are the functional requirements of the project?
2. General requirements: What is needed to support deliverables, things like project management, meetings, spec docs, etc.?
3. Costs: What is the project budget and how will it be allocated?
4. Timelines: Are there any key deadlines that must be met?
5. Design: How will the project be designed and what technologies will be used?
6. Resources: What personnel, hardware, and software resources are available for the project?
7. Risks: What risks are associated with the project and how will they be managed?
8. Testing: How will the project be tested and what criteria will be used to evaluate quality?
9. Maintenance: What processes and procedures will be used to ensure the project remains up-to-date and secure?

Asking critical questions - and lots of them - is the key to uncovering and understanding expectations. Every project is different, so there isn't a one size fits all list of questions you should ask. The goal of your questions is always to add details to the project to remove any assumptions. Challenge your own assumptions about a project and write them down. Consider what might be necessary from a business perspective, a solution perspective, and a stakeholder perspective. The Dabble Lab maintainer is taking responsibility for the success of the project, and must have a full understanding of the, what, whys, and how's of the project.

The person leading the discovery process needs to be experienced enough to ask the right questions. Depending on the complexity of the project, you might need multiple people with experience in different areas to ask all the necessary questions. But asking the right questions is critical to an effective discovery process and ultimately to the success of the project. The project planner's first, and arguably most important responsibility, is to ensure the discovery process is done thoroughly by qualified individuals. As a result of the discovery process, you should have a clear understanding of stakeholder expectations and be ready to define clear goals and objectives.

## Mission, Goals, and Objectives

A project's mission defines its overarching purpose. It answers the question: Why are we doing this? The mission should serve as a "North Star" for goals and provide the general purpose for the project. For example:

*Our project mission is to delight customers with an automated customer support solution that is scalable and cost-effective to implement and maintain.*

Goals and objectives should always be aligned with the project's mission, but they are more specific outcomes the project is trying to achieve. At Dabble Lab, we define goals as things you want to accomplish that are not fully under your control, while objectives are things you want to accomplish that are completely in your control.

_____graphic

- Goals: Things you want to accomplish that are not fully under your control
- Objectives: Things you want to accomplish that are 100% under your control

_____

Goals and objectives should always be SMART—specific, measurable, achievable, relevant, and time-bound. Here is an example of a SMART goal:

*By the end of this calendar year, increase customer satisfaction scores by 15% and reduce support center costs by 20% with an improved AI-driven customer support system.*

If customer satisfaction scores have increased by 15% and costs have decreased by 20% by the end of the calendar year as a result of an AI-driven system - the goal has been accomplished. There is nothing opinionated about it. Consider the following example goals:

**Goal (take 1):** Improve customer satisfaction with an AI system that accurately answers customers questions when human agents aren't available.

While this goal might seem clear, the outcome can't be determined. Without defining how customer satisfaction is measured, or what the current level of customer satisfaction is, there is no way to know if satisfaction has improved. It's also not time bound. So, we're not clear on when the goal should be achieved. Let's take a look at another example.

**Goal (take 2):** By July 1st of next year, implement an AI system that can accurately answer customer questions when human agents are not available.

This goal is much better than the previous example. If the system answers questions accurately when human agents are not available, the goal has been achieved. But wait, what questions will it have to answer accurately? All questions? Is that realistic? Also, does the term "implement" mean the system should be in full production or just working at some level? Let's try again, this goal isn't as clear as it could be.

**Goal (take 3):** By July 1st of next year, be in production with an AI system that can accurately answer our top 20 most frequently asked customer questions when human agents are not available.

The final version of the goal is pretty well written. You'd still want to clarify in your plan exactly how you'll go about testing to ensure that the 20 questions are being accurately answered. But that level of detail is not necessary when defining goals and objectives as long as it's clearly defined in the plan.

## Opinions Matter

You might be wondering about cases when the outcome of a goal can't be objectively determined. Are there ever cases when it's acceptable for goal or objective outcomes to be a matter of opinion? Yes, there is. But you should make it clear whose opinions matter. Here's an example:

**Objective (take 1):** The app should include a super simple registration and onboarding process.

Defining "super simple" is the challenge. But also, we need a way to determine if and when this objective can be considered complete. The best way to do this is to define who will ultimately be responsible for deciding when it's done. So after we know who will be responsible for signing off on the completion of this objective we'd rewrite it in the following way.

**Objective (take 2):** The registration and onboarding process should pass a final review from Jane Smith before going to production.

In this case, the objective has been completed when Jane says it has. Other opinions don't matter. The key is choosing just one person that has the authority to make the final decision. That person might need to consult with others to make their decision, but only one person should have the final say.

## Start Your Discovery

Usually, we start with a limited client engagement just for conducting the initial project discovery. Understanding the scope as much as possible from the start is necessary to reduce long-term margin of error. We don't start a project until expectations are clear, and they must be written down. Even small details that seem insignificant can easily contribute to misunderstandings, not to mention, people's memories are far from perfect. Project discovery is on-going, and details should be written down and kept wherever the project maintainer sees fit. Maintainers then translate expectations into requirements.

# Chapter 3: Requirements

While expectations can be written in many different ways, requirements follow specific guidelines. In Dabble Lab's workflow, requirements must be client-verifiable and should not exceed 8 hours to complete. Requirements fall into two categories: functional and general. Functional requirements define project deliverables, while general requirements are non-deliverable activities that are still needed to complete the project—examples include project management, meetings, and defining project guidelines. Technical tasks and work items aren't requirements because a client without a technical background cannot see them to verify, they are complete.

Here's an example:

–

Expectation: By July 1st of next year, be in production with an AI system that can accurately answer our top 20 most frequently asked customer questions when human agents are not available.

Example General Requirement: Through July 1st, hold a weekly, 30-minute meeting to update stakeholders on the status of the project.

Example General Requirement: By December 15, present a System Design and project spec doc to Jennifer.

Example Functional Requirement: By January 15th, have an SMS interface for our AI system that can respond to one or more questions as described in the System Design doc.

Example Functional Requirement: By February 1st, have the bot answering all 20 FAQs via the SMS interface.

–

Notice that while our expectation is broad, our requirements are zoomed into the details without getting too technical for anyone to understand. For functional requirements missing technical details, there are general requirements to have documentation with technical specifications signed off by qualified individuals—and this same approach can be applied to budgets, resources, maintenance plans, testing, and so on.

## Example Functional Requirements

Like goals and objectives, the key to defining functional requirements is to write them so that they can be objectively completed (SMART). Here are some examples of poorly written functional requirements and how they could be rewritten in a more objective way.

Example 1:

- Subjective (wrong way): The app must have a well-designed user interface.
- Objective (correct way): The Material Design User Interface Guidelines should be followed throughout the application.

Example 2:

- Subjective (wrong way): The app must be scalable.
- Objective (correct way): The app must scale to support 1,000 concurrent users with less than a 200ms response time, as measured by Apache JMeter.

Example 3:

- Subjective (wrong way): The app must be user friendly.
- Objective (correct way): John Doe must sign off on the app's usability before any version ships to production.

Defining functional requirements can be tedious and time consuming. But shortcutting this process is much more costly and time consuming in the long run. You most likely won't be able to anticipate every functional requirement up-front. But that should not be for a lack of trying.

## General Requirements

In addition to functional requirements, there are general requirements to consider. General requirements are not deliverables, rather necessities to achieve the functional requirements. While every project is different, some common general requirements include project planning, project management, and project slippage.

### Project Planning

Planning is necessary for every project. The amount of planning needed depends on the complexity of the project, but the number one reason projects fail can almost always be traced back to poor planning. While the time needed might differ from one project to the next, every project will have a planning requirement.

### Project Management

Like planning, every project will require time for project management. Even if the project doesn't have a formal project manager, the time spent understanding, communicating, and prioritizing project requirements could all be considered project management.

### Project Slippage

The planned and actual time required to complete a project will almost never be exactly the same. To plan for this, project slippage needs to be considered a requirement. Project slippage is a key consideration in estimating which we'll discuss later. But for now, you can think of it as a requirement to account for planning miscalculations - like a margin of error.

## Unknown Requirements

While ideally you can anticipate all of a project's functional and general requirements before the project begins - that's rarely the case. Unanticipated requirements will likely come up as the project gets underway. Requirements that get added as the project progresses are a normal part of software development. In fact, most modern software development follows an agile development methodology that expects requirements to be added over time.

## Requirements Summary

Following your initial project discovery process, translate stakeholder expectations for both general and specific project requirements, and write them in a way that ensures they can be objectively evaluated as done/delivered. If there could be any disagreement as to whether a requirement was completed, it was not well defined. This is critically important because without objective requirements, it's impossible to define a budget or delivery timeline.

# Chapter 4: Budgets and Timelines

Software estimating is the process of determining the effort, duration, and cost required to develop a software product. It is a crucial part of software project planning and management because it's necessary for allocating resources and monitoring the progress and status of a project.

There are several techniques that can be used for software estimating, including:

- Expert judgment: Utilizing the experience and knowledge of a team of experts to estimate project parameters such as effort, duration, and cost.
- Analogous estimation: Using the parameters of a previous, similar project as a basis for estimating the parameters of a new project.
- Parametric estimation: Using mathematical models and statistical data to estimate project parameters.
- Three-point estimation: Using a combination of best-case, most likely, and worst-case scenarios to estimate project parameters.
- Planning poker: A consensus-based technique where team members use cards to estimate the effort required for a task.

No matter what technique is used, estimating is inherently uncertain and subject to change as the project progresses. This creates a great deal of risk that needs to be carefully considered and managed. Budget and timeline expectations depend on clearly defined requirements. But even with clear requirements, setting expectations for timelines and budgets can be tricky. Start with the assumption that there won't be enough time or budget to complete all of the requirements and focus on prioritizing.

## Prioritizing Requirements

To prioritize requirements, ask the following questions:

1. What are the top three requirements that need to be delivered?
2. What dependencies exist between requirements?
3. Which dependency requirements need to be completed to deliver the top three?

After answering those three questions, use those requirements to start estimating the budgets and delivery timelines. Do that by estimating the hours needed and an estimated delivery date for each requirement.

## Estimating a Project

Imagine you have a project with 100 requirements. Suppose you estimate that each requirement will take 10 hours to complete. Given that, the project should be completed in 1,000 hours. But in addition to the hours required (aka: effort) you'll also want to estimate the project duration and costs.

Let's assume for our example that the cost will be $100 per hour and that we'll have two dedicated people working on the project. Given this, the project cost estimate would be $100,000, and since we have two full-time people (40 hours per week each) working, the project is estimated to take 25 weeks to finish.

Each week, we should have 40 requirements completed and be 1/25 closer to completing the project. But what if after the second week we've only completed a total of 20 requirements? Well now it's looking like the project will take 26 weeks and require 1,040 hours—and that's assuming all the other requirements get delivered on time. But if the first two weeks are an indicator of how the following weeks will go, the project will take 50 weeks, and require 2,000 hours to complete. If that's the case, the project costs and timeline will end up being 100% more than we'd estimated. In our example, the good news is that we are seeing the this after week two. So, we might be able to get back on track. But is the issue an estimating problem or something else, and either way, what do we do about it?

## Course Correcting

The more complex a project is the harder it will be to estimate budget requirements and delivery timelines. Course correcting is about adjusting expectations, priorities, and resources based on estimated versus actual costs and delivery dates.

First, we want to understand why our estimates are off. The two most common reasons are unclear requirements and inaccurate estimates. In either case, the responsibility falls on the project maintainer. If requirements are unclear, they should review the requirements for writing requirements and work with other Dabble Lab maintainers to ensure their requirements are properly defined.

If the requirements are clear but the estimates are inaccurate, the maintainer must reconsider if the person or persons doing the estimating are fully qualified. Note that even people who have been building software for decades can be inexperienced with a particular technology—maintaining robust professional relationships with project team members is important for a maintainer to understand team members' strengths and weaknesses.

Once the reason for the bad estimates is resolved, we have to correct our mistakes. How we achieve this depends on the project and its constraints.

If we are under-budget, we can add more time. While we spend more hours, that can be ok if the client has already paid for them and doesn't have anything else to spend the hours on. We can keep the project on schedule by adding more people to the project, so more hours are spent each day.

If we can't spend more hours, we can reorganize the people working on the project. Swapping out less experienced Dabblers with more experienced ones can help us catch up, but our ability to do this depends on our other active projects.

Finally, in a worse-case scenario, we cut back requirements. Generally, as long as we address this change well before the delivery date, clients understand that estimating complex projects is difficult. Because we estimate at the requirement level, we notice if our estimates are off pretty quickly into a project, and we are able to address concerns with the client proactively.

# Chapter 5: Where to Go from Here

Project planning is an ongoing process, led by Dabble Lab maintainers. At the core of this process is the idea that nothing can be assumed. All expectations and requirements must be written down in a way that success can be measured objectively by anyone—regardless of technical background.

We know project planning isn't easy—it takes practice and skill, but it's one of Dabble Lab's key value propositions.