CHAPTER 3

Identifying the Problem & Designing the Solution

Every computer system is (or at least should be) the result of solving some type of problem. While “Hello World” apps are great, I firmly believe that the best way to learn any technology is to apply it to a real problem and see how things really work.

We’re going to be adhering to that principle throughout this book. This chapter discusses a very common problem in most organizations that can be solved technically. You’ll also look at some of the specific things you need to consider when designing web-based systems in general and APEX specifically.

Identifying System Requirements

Almost every company, no matter the size, will at some point need to implement some sort of help desk. Whether it’s an internal one to track employee questions and problems or an external one to track client issues with commercial software or hardware, the basics of a help desk system are fairly standard.

Most help desk systems are driven by the notion of a “Trouble Ticket” or “Ticket.” This term is a leftover from the days before computers: most problems were reported over the phone, and troubleshooters used a physical paper ticket to log a call. The information contained on that paper ticket included a description of the problem, the person having the problem, when the problem was logged, etc. Then, throughout the process of troubleshooting and, hopefully, solving the problem, the engineers would write down each step of the process and include any documentation of the problem they gathered along the way. Today, it would be very surprising to see a help desk system that wasn’t computerized, even if it’s only a spreadsheet of issues with notes and statuses.

You’re going to attack the help desk system with APEX. Before you dive in, you need to clearly understand the problems you are trying to solve. If nothing else, you need to review the current system.

Never a Clean Slate

Almost no computer system written today starts from scratch. There is almost always something in place, even if it’s just some loose guidelines or ideas.

For this example, let’s say that your company has a very basic system in place, but it’s no longer meeting the needs of your growing user community. Your goal is to create a new system that will make the logging of issues and their solutions much easier for everyone involved; however, to do that, you must understand the needs of the users and the functionality of the system that is in place now.
A Broken System

In general, the users of help desk systems can be categorized into two groups: people who log problems (end users) and people who help solve the problems (technicians). Depending on which user community you fall into, it’s likely you will have different needs, but overall, the system should help the end users and the technicians communicate about the problem or issue.

The first step is to understand how your help desk is being managed today and why it’s not working. Speaking to both the technicians and the end users can provide a huge amount of information, but the challenge is that this information usually comes in the form of complaints about the current system.

Quizzing the end users reveals that their main complaint is that they never know the status of the problems that they’ve logged. They can go days, sometimes weeks, without communication from the technicians, and in the eyes of the user, no communication means that no one is working on their problem. Another user complaint is that the help desk technicians often don’t know how to contact them to ask further questions or communicate progress.

On the other end of the issue, the technicians are overloaded. Originally, the help desk was only one person, and ticket information was kept in an Excel spreadsheet. Now there are several technicians working independently. While performing their daily duties, each needs to update the spreadsheet with information regarding the tickets assigned to him. The increasing number of people accessing a single spreadsheet causes problems, as only one person is able to open and update the spreadsheet at any given time. The technicians are also tired of constantly being called by users wanting an update on the status of their issues.

It’s obvious that the system is broken. Neither the users nor the technicians are happy about the situation. It’s your job to take the information you’ve gleaned from these conversations and design something that will address the needs of both user communities.

How Do You Fix Things?

With the information you’ve gathered so far, you can now define some loose requirements and break them down by user type so you can have a much clearer understanding of what each community needs. Then, from those requirements, you can begin to think about the database design that you’ll need to create in support of them.

Defining the Requirements

You can look at requirements from two perspectives. End users will have one set of requirements and technicians another. Some requirements will overlap between the two groups. Others will be unique to one group or the other.

End users should be able to:

- Create a new ticket outlining their problem.
- See the status and progress of tickets.

Technicians should be able to:

- Easily identify and view new tickets.
- Easily identify which tickets are directly assigned to them.
- Search existing tickets.
- Create new tickets on behalf of an end user.