STEP-BY-STEP WEB APP DESIGN

BY ABHIJIT RAWOOL

TABLE OF CONTENTS

CHAPTER 3

UNDERSTAND REQUIREMENTS

Understanding the requirements of a customer is by far the most important activity in the entire product development cycle.

I think I don't have to tell you why.

Still, many people fail to understand the requirements correctly.

Why does this happen? Let's see.

MISCONCEPTIONS ABOUT REQUIREMENTS

Let me clear up two common misconceptions about requirements before I start discussing how you can fully understand the actual requirements that your customers will have from your web application.

REQUIREMENTS ARE NOT FEATURES

Everyone views the Requirements Gathering phase as something in which they have to list out all the possible features that a prospective customer wants.

Unfortunately, they do not understand that Requirements and Features are two different things.

A requirement is something that your customer needs, while a feature is something that will eventually fulfill that need.

It is very essential that you do not think about potential features when you are gathering requirements.

If you do that, then you are trying to find a cure before you have even understood the disease. Don’t do that.

Understand the requirements first. Once you have understood the requirements, you can then brainstorm all possible features that can fulfill those requirements.

REQUIREMENTS ARE NOT PROBLEMS ALONE

If you have been designing web applications for any length of time, you would have heard people say that you need to understand the problems your customers are facing and then build your web application to solve those problems.

These problems eventually become requirements for your web applications. That should not happen.

Problems do create some of the requirements, but not all. Understanding problems is just one part of understanding the requirements.

To fully understand the requirements, you need to understand the following 3 things:

1. Understand the Problem
2. Understand the User
3. Understand the User's Tasks

It is a common practice to start defining the requirements after understanding the problem. Understanding the users and the tasks carried out by the users is often left to the Design phase.

However, in my experience, understanding the users and their tasks help to properly define the requirements. I am not asking you do extensive research on the users and the tasks, but you should at least have a good idea of who your users are and how they perform the tasks that you will eventually support in your web application. More on that in a moment.

UNDERSTAND THE PROBLEM

Your primary aim in the Requirements Gathering phase should be to understand the problem the users are facing, the problem they want to solve by investing in your web application.

Every business that wants to buy an application is facing some problem and they will invest in your web application if you can solve that problem for them.

Let me try to explain this with a couple of examples. Say you hear the following two requirements from your customers:

1. We want to digitize all of our physical documents,
2. We want to sell tickets for our events to our employees online.

At first glance, both of these requirements seem complete, but if you take a closer look at each of these, you will realize that none of them tell you what problem the customer is trying to solve.

The customer might want to have all the files in digital format because they are finding it difficult to manually track all the files pertaining to a project. Maybe they are facing a problem to keep all the teams in sync with one another because the documents have to be physically moved from one team to another every time they are updated.

Once you understand the problem the customer is trying to solve, you can easily see that just digitizing the documents is not going to help. You may have to provide a way for the users to easily group and share relevant documents with one another.

Similarly, the requirement to sell event tickets online does not tell you anything about the problem being faced. The customers might be thinking that by letting the employees buy the tickets online, they will be able to sell more tickets than the number of tickets they are selling today over the counter. But in reality, maybe every employee is not even aware of all the events taking place.

In such a case, a better solution is to build something that can effectively make the employees aware of the events taking place than an application to sell tickets online.

These are just a couple of requirements that never told you anything about the intentions (problems) behind stating these requirements.

While gathering requirements, if you just list the requirements without getting down to the actual problem, you will end up designing an application that no one wants to use.

Quite often,

The requirement that your customer's state is actually not what they want.

To find the problems being faced by your customers, you can start by asking "why" for every requirement that you hear.

- Why do you need this feature?
- Why do you do things in this way?
- Why do you need this to happen?

Questions like these will help you get to the root of the problem and find out requirements that you would have otherwise missed if you had just concentrated on creating a list of requirements.

UNDERSTANDING THE PROBLEM WHEN YOU DON'T HAVE ANY CUSTOMERS

If you are thinking of launching a web application then most likely you don't have customers yet.

I assume you have already found the problem that you want to solve with your web application. So now the question is, how do you get to the bottom of understanding the problem?

Well, the most reliable way to do this is to talk to prospective customers. However, this is easier said than done.

Why would customers waste their time giving you feedback on your web application when they can instead spend time doing something that is valuable to them?

The answer is simple. Make giving their feedback valuable for them by offering your web application for free when you launch. You can even think of giving them a huge discount.

It works something similar to beta-testing where you invite people to test your application and in return, you offer them something.

Only, in this case, your aim is to understand the problem at a granular level rather than test your web application.

However, here is the catch. Don't contact people who don't want or have the problem your web application is trying to solve.

Whenever you meet prospective customers, try to figure out if they really are facing that particular problem and if they want a way to solve it. Are they already using some other web application to solve that problem? If so, are they happy with that web application?

The aim of doing all of this is not to get new customers. Drill that into your head.

The aim is to find customers who have that problem and get to understand the problem better.

Even if you can get 2-3 customers to give you meaningful insight into the problem, that will greatly help you figure out the real requirements customers will have from your web application and what features you can build to satisfy those requirements.

UNDERSTANDING THE PROBLEM WHEN YOU HAVE CUSTOMERS

If you already have customers or clients for whom you are building a web application then your task of understanding the problem becomes a lot easier.

You can just talk to the end users who will eventually use your web application day in and day out to understand the problem. We will discuss this in great detail in the next section.

If you already have customers who are using your application then just get in touch with them and ask if your web application really solves their problem, and if not then try to understand why it is not able to solve the problem.

Again, the aim is to understand the problem, not to figure out the list of features or bugs in your web application.

Just concentrate on discussing the problem as it is stands today and not discussing possible solutions to the problem.

NEXT STEP

I hope you found this sample from Step-by-Step Web App Design book useful.

If you would like to read the rest of the book then you can buy it from here:

https://stepbystepwebappdesign.com

Your purchase is backed by 100% 30-day Money Back Guarantee. If you don't like the book or feel this book wasn't for you, there are no hard feelings! Simply send me an email at abhijit@webapphuddle.com and I'll immediately refund your money.