**Identifying Software Requirements**

Do not overpromise and/or overestimate your ability.

The resources the clients have determines how amazing the end product can be, they might want more than they can afford.

You must also make sure the clients want/idea meets their need. They might need something much less fancy to fix their problem.

**Two types of requirements NBNBNB**

**Functional Requirement:** The requirements needed for the system to work properly. eg Facebook profile creation. Ie if these requirements aren’t there, then Facebook is not Facebook.

**Non-Functional Requirement:** The requirements not needed for the system to work properly, but improve the product/user experience in some way. Eg Dark Mode. They describe how functional requirements may work; they effect quality a lot.

Non-Functional Requirements should be categorized. It must be **quantifiable**. ie User-Experience, Safety/security.

**--New stuff below**

**Eg** Functional Requirements of a Car: These are any key things to complete the requirements.

Transport you from A to B  
  
**Eg** Non-Functional Requirements of a Car

Top Speed; Tinted Windows; Fancy Wheels; A seatbelt is non-functional, but is needed for the system(car) to work.  
-These do not impact **that** we got from A to B, but impacts **how** we got from A to B.  
  
**Therefore** Non-requirements do not effect the fact that X functional requirement is done, but how it is.

**Do not be too broad.** Be **specific** in what the non-functional requirement must do.  
Also do not say Eg Reliable: The system must be reliable.  
Instead Eg Reliable: The system must have an error rate of 1 or less per 100 days. So that the IT guy only has to be brought in every 3rd month. (Make sure you mention the quantified thing being measured)  
  
If the requirement is very much needed for the product to run, but is not listed, it is still a non-functional requirement.

--**New stuff above**

**A use case**

**The Swift System** (Not in Textbook)

The relationship with the sponsor

1. Inception:
2. When the relationship is being formed, when a client has a problem and you can provide the solution (you approaching them or vise versa).
3. It is important to find out who all the stakeholders are at this stage. (asking the right people the right questions
4. Elicitation: When you get more information from the client about their problem/idea. Getting a better understanding of the business goals (must be honest).
5. Elaboration: Creation of a model of requirements (These are in chapter 8). You are elaborating on the requirements given by the stakeholders.
6. Negotiation: He stakeholders discuss between each other with what is the most important things that the program must have/do. What must be prioritized.
7. Specification: Now you show the client your understanding of what they want in a standard and consistent way ie Written Document.
8. Validation:   
   a. You need to validate/test the specification document/other work products produced for quality.
9. Normally the person who wrote the document does not do this.
10. Examines the specification faze to make sure there is no ambiguity, errors.
11. The work product must conform to the business process and the product.

Feasibility study: Cost of developers, cost to develop app, if APIs need to be bought, special equipment.  
Things like uber, computers ect must be part of the developers hourly rate. Your rate must be, at least, the same as a junior software developer!

Make sure your requirements for the project are doable and relate to what you’re doing. They must be quantifiable (can be measured in some way)

**The Foundation (Text book: Establishing the Foundation)**

How do we carry out the above.

What is a stakeholder: Those who are both directly and indirectly affected by the system.

After speaking to a stakeholder, it is important to ask is there anyone else you need to talk to.

Step 1: Identifying stakeholders.

Step 2: Recognizing multiple view points. Eg Marketing and accounting people will have different interests. (The 2 faces image) Your job is to take all these viewpoints/requirements from the stakeholders and unify them into one project/system.  
  
For Requirements:  
Very Dangerous: Unspoken assumptions.   
Team members interpreting things differently for the same system(communication key).  
Requirements being stated vaguely/in a way that is difficult to verify

Step 3: Working towards Collaboration. Different people working on the same project leads to different opinions.

So for collaboration to occur, you must:

1. You need to identify all the areas of communality.
2. You need to identify all the areas of contention.
3. IE, What all the stakeholders agree on and you write down to identify areas of conflict. To negotiate the end result.
4. It is important to have a leader who can make the final discissions after the negotiation.

Step 4: Asking the first questions. This happens during Inception. Finding out who the projects benefits. Who is behind the request for the work. Who is going to use the solution. What is the economic benefit for the solution. DO not build a system that is already there. The questions need to be context free (not technical).

These questions are to identify all the stakeholders in the projects, and the benefits a successful implementation will have on them.

Now, the following questions help the customer find any issues/problems (the customer’s needs)  
  
How would you categories good output/solution.  
What problems would this solution address.  
Can you show me the business environment that the project is for.  
What special performance issues will effect the way the system building is approached.  
  
Find out if the customer feels like they have been able to properly able to convey their needs:   
  
Are you the right person for these questions  
Are these questions relevant  
Am I asking too many questions  
Are there any questions I should Ask

Step 5: Traceability. Your requirements must always be traceable throughout the system (Through all the requirements, diagrams, prototypes ect). Must be able to look at final system and trace it to the origin requirements (it must make sense, very important). Verification and Validation is part of traceability: Is the system doing what it is supposed to do, and Is it doing it right.

Step 6 (not in text book): 3 Diseases Software engineers can have.  
a. Featuritus: When you add features that don’t solve the problem.  
b. Flexibilities: When there are too many things to do one thing, but nothing happens after (doesn’t actually solve the problem)  
c. Performitis: When too much time is optimizing code/perfecting code.