User Stories to Project Initial Architectural Overview

The current contents you've developed almost comprise your SRS. In the evolution from the original description to now, things may have gotten out of sync. The assumption is that the project description, user stories, and requirements are complete and correct at this point. If this is not the case, continue to work on them. We have nomore iterations dedicated to that part.

Now you're assembling an architectural overview to complete the SRS as follows.

Each user story gets a separate diagram. The flow through the diagram reflects the execution of the user story.For example, consider this user story:

As a student I want to post my assignment so it can be graded.

Add components that you already know you need, like a webserver. Others you can add as you encounter them. So in this case, the mental flow could look somewhat like this (W5H *who*, *what*, *where*,*when*, *why*, *how*). You don't have to write it out, but you do have to be able to articulate it when asked.

What is flowing? The assignment.

What's the form of an assignment? A file.

Where does the flow start? On the webpage in a browser on the client's machine.

Who does the submission? The student.

How do they post an assignment? Lower-level details aren't necessary yet, like the web form. It's enough to say they use a control element to find the file to upload, which is somewhere on their filesystem. This introduces their client filesystem into the diagram. It also introduces a clear boundary between their side (the client filesystem)and the other side (the webserver).

What happens next? The file magically uploads by following an arrow (labeled “file”) from the filesystem to thewebserver, but where to then? There’s probably a server filesystem needed, and an arrow to it.

What happens next? There’s probably a record of the submission stored in a database, so introduce the database with another arrow from the webserver. But on this arrow it’s the record that moves, so label it so. You don’t have to detail what’s in the record as long as it’s clear in your mind that everything you need to store in the database is available in that unit that’s following the arrow. The webserver must produce this output; the databasemust accept this input.

Keep this up until every aspect of the user story is captured. Therefore user stories shouldn’t be large. If they are,they probably should be broken into multiple smaller ones.

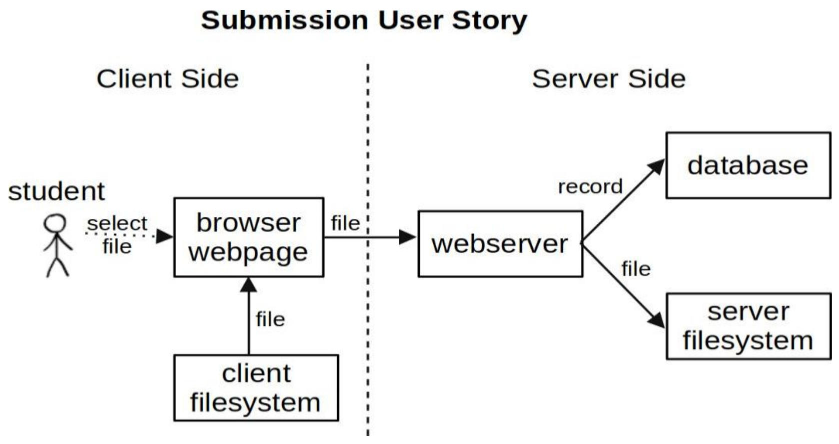
You can envision there might also be an confirmation email going back to the student through a mail server to a mail client (although this would probably be a separate user story because other features likely use it, too).

The general form of a flow diagram is an input-processing-output chain or network. Each component is a processing box with input(s) and output(s) (although it’s possible to lack either or both). The output(s) become the input(s) to one or more other components until the end of the chain. The processing is magic, as long as it’s clear to you what’s going on at this point, but you can put some basic description in the box if necessary. At minimum it needs a meaningful label.
Make the diagram readable, generally from left to right, but don’t get fancy because this is a draft. Indexing isexplained below to make it easier to keep track of things.

Once you're done with the individual diagrams, create a unionized one with all the components and arrows. The data won't be indicated on the arrows because you're not executing a particular user story. This is the architectural overview of the capabilities the system provides. The perimeter is the implied scope of the expected implementation. You should be able to quiz yourself by executing any story on this one without looking at the original one.

If you run into issues, like not knowing how to represent something, it's probably an indication that this is a trouble spot. Do your best to explain what's supposed to happen here, and we'll work through it next week.

Here's a possible diagram for the example above (minus the sequencing):

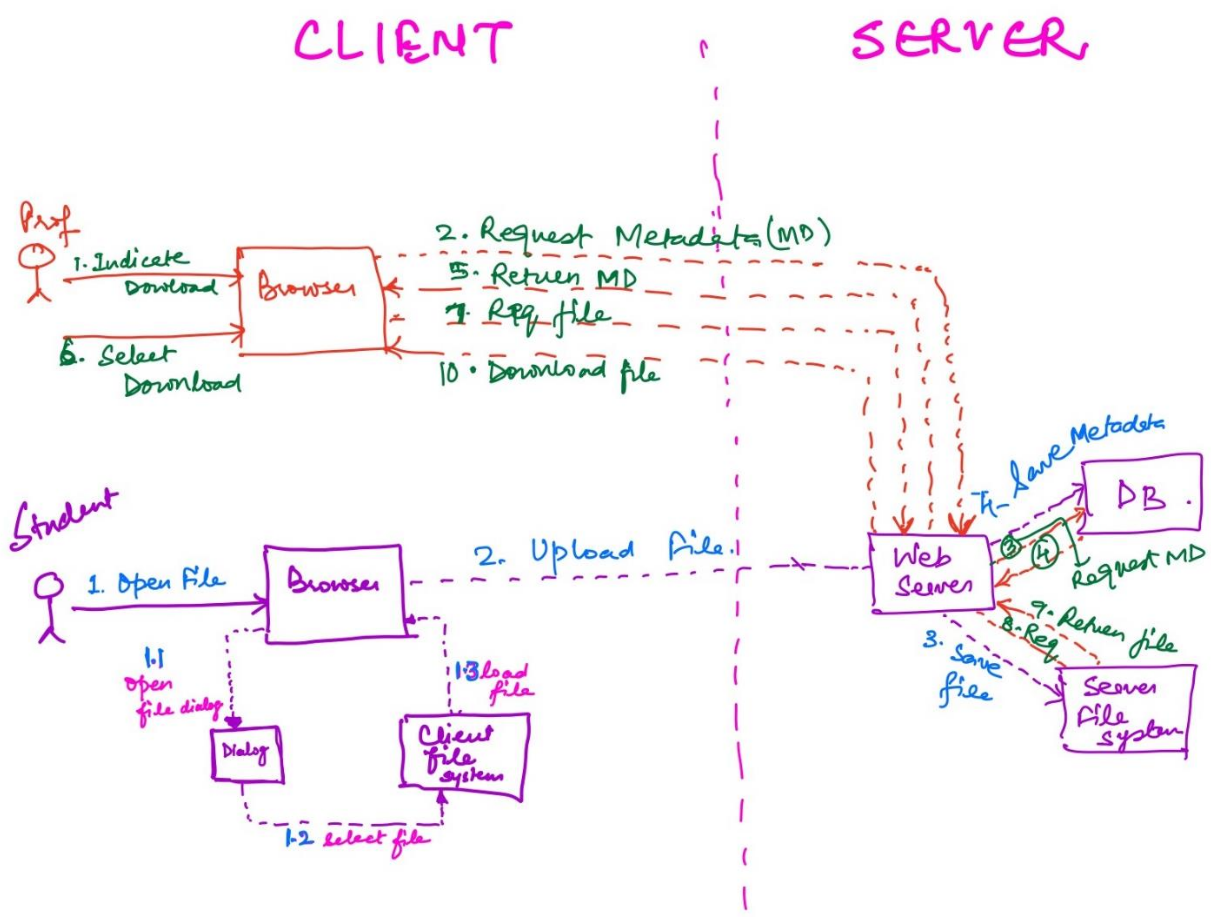


Note that our user story has the benefit "so it can be graded," but this is not captured here. It would be a different user story with additional components to address the grading part from the professor's perspective. Make sure the stories are complete, correct, and consistent, as well as their corresponding diagrams.

Make sure the architectural overview diagrams capture the sequencing. Put a number on each arc. In some cases more than one order may be possible. It's ok to indicate just one (like the file path then the record path, or vice versa). If there's not a linear flow for some reason, explain what's going on. For example, step 1 is from box A to

B, then step 2 is from C to B, and finally step 3 is B to D. Explain how C happens. You might be missing an arc to activate it, even if there's no data going across the arc.

The example can be seen below



Finally, use everything in each user story to state under it the requirements and possibly the specifications for it.

Requirement: The form of an acceptable solution. For example, There shall be a feature to print the document.
Specification: A constraint on the requirement. For example, The print feature shall use the common print dialog provided by the operating system.

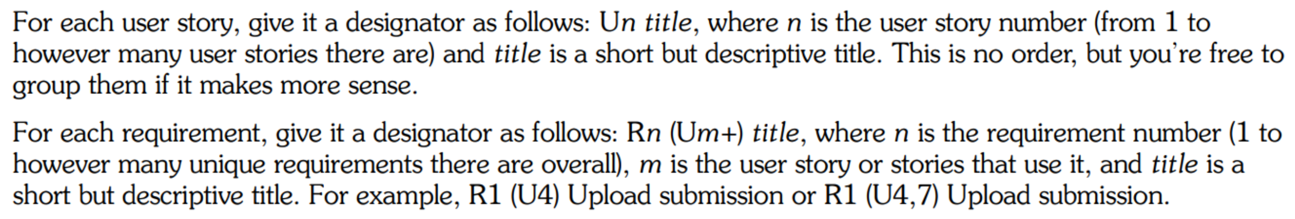
A combined requirement with Specification can be like: *There shall be a feature to print the document. It shall use the common print dialog provided by the operating system*

Or

There shall be a feature to print the document that uses the common print dialog provided by the operating system.

Remember, requirements and specifications must be in a form that can be answered with a Yes/No question as part of the verification process.

For example, is there a print feature that uses the common print dialog provided by the operating system? You don't need to include this part in the task, but make sure you at least articulate it to yourself and your team such that everyone agrees the form is correct



For each user story, give it a designator as follows: U*n* *title*, where *n* is the user story number (from 1 to however many user stories there are) and *title* is a short but descriptive title. This is no order, but you're free to group them if it makes more sense.

For each requirement, give it a designator as follows: R*n* (U*m*+) *title*, where *n* is the requirement number (1 to however many unique requirements there are overall), *m* is the user story or stories that use it, and *title* is a short but descriptive title. For example, R1 (U4) Upload submission or R1 (U4,7) Upload submission.