A design doc — also known as a technical spec — is a description of how you plan to solve a problem.

**A design doc is the most useful tool for making sure the right work gets done.**

The main goal of a design doc is to make you more effective by forcing you to think through the design and gather feedback from others. People often think the point of a design doc is to to teach others about some system or serve as documentation later on. While those can be beneficial side effects, they are **not** the goal in and of themselves.

A design doc describes the solution to a problem. Since the nature of each problem is different, naturally you’d want to structure your design doc differently.

#### **Title and People**

Hospital Management System

Viktor Lazarev

Trevor Strikeleather

Cristina Samford

#### **Overview**

The hospital management system will allow hospital staff to store, edit, and retrieve patient medical records. The staff will have the ability to add a new patient to the system, edit and/or update patient information, create new medical appointments for patients, and generate a report that will display current and previous appointments.

#### **Context**

A hospital Management System is an integrated software that creates user friendly solutions for medical staff which will allow a hospital to meet state and Federal guidelines for patient record processing. This software provides the company with an efficient automation of required administrative, legal, and financial tasks.

#### **Goals**

* Fully functioning user menu
* Patient module that allows the user to create a new patient record, edit patient record information, and delete a patient record
* Appointment module that allows the user to create a new appointment, view a report with current appointments by patient, and delete an appointment.

**Non Goals**

* Creating a user profile for patients to manage their own appointments.
* Generate billing reports compiled from patient records.
* User profiles for other medical staff

#### **Milestones**

#### Start Date: July 16, 2020

Milestone 1 — Create Requirements document-First Draft

Milestone 2 - Create Design document

Milestone 3 - Create User Menu

Milestone 4 - Create Patient Module

Milestone 5 - Create Appointment Module

End Date: August 5th, 2020

#### **Existing Solution**

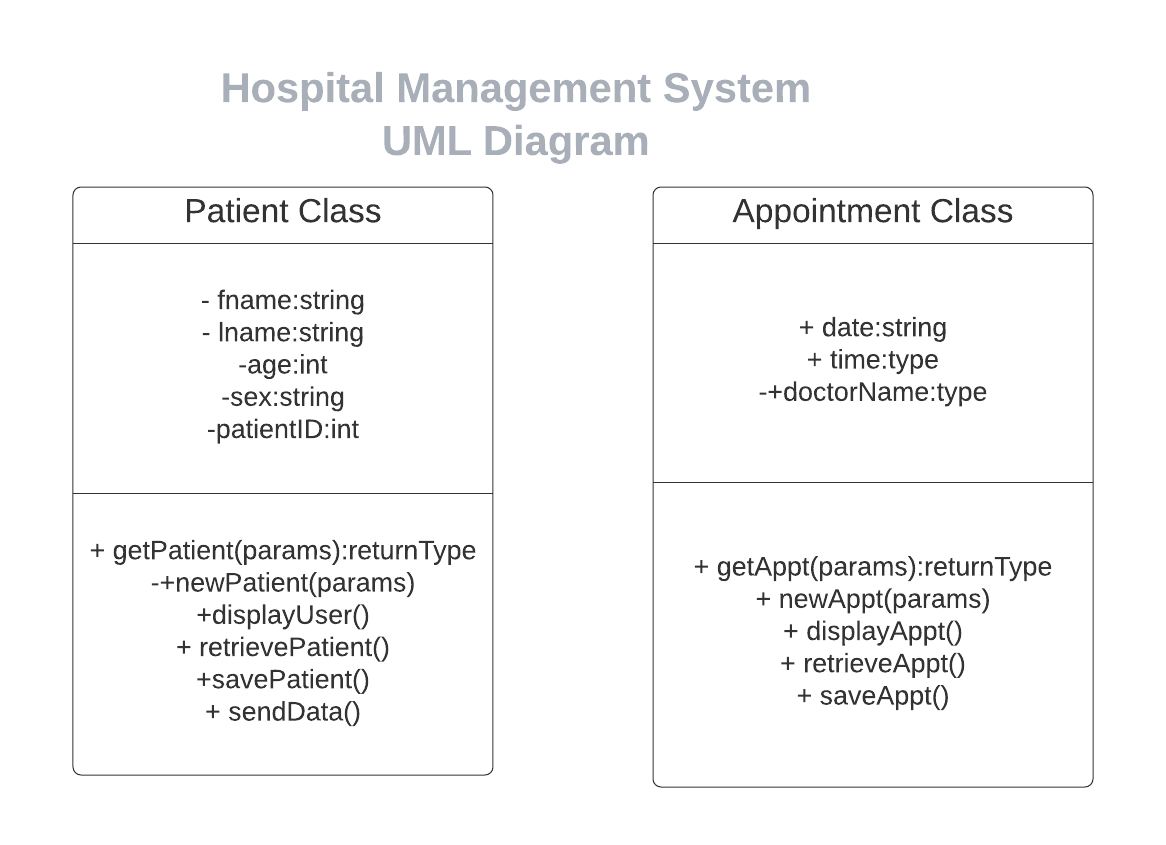
In addition to describing the current implementation, you should also walk through a high level example flow to illustrate how users interact with this system and/or how data flow through it.

A **user****story**is a great way to frame this. Keep in mind that your system might have different types of users with different use cases.

Currently the hospital’s system requires administrative staff to manage medical appointments manually by writing the required information in a book. This has become problematic overtime due to the time required to retrieve information for a specific patient. The current process for patient records is to document demographic information and patient care details in only a physical file. While required to keep physical medical records by law, this process without automation and digital records creates a time-consuming task for administrative personnel when retrieving patient information.

#### **Proposed Solution**

Some people call this the **Technical Architecture** section. Again, try to walk through a user story to concretize this. Feel free to include many sub-sections and diagrams.



Edit UML Diagram Here: <https://app.lucidchart.com/invitations/accept/83872f83-f6e9-479f-9cf1-155cdb6472d1>

Provide a big picture first, then fill in *lots*ofdetails. Aim for a world where you can write this, then take a vacation on some deserted island, and another engineer on the team can just read it and implement the solution as you described.

#### **Alternative Solutions**

What else did you consider when coming up with the solution above? What are the pros and cons of the alternatives? Have you considered buying a 3rd-party solution — or using an open source one — that solves this problem as opposed to building your own?

#### **Testability, Monitoring and Alerting**

* Verification that all required fields are used when the user adds a new patient record.
* Verification that the user is able to login with correct username and password.
* Verification that the correct message displays if the username and/or password do not meet requirements.

#### **Cross-Team Impact**

How will this increase on call and dev-ops burden?

How much money will it cost?

Does it cause any latency regression to the system?

Does it expose any security vulnerabilities?

What are some negative consequences and side effects?

How might the support team communicate this to the customers?

#### **Known unknowns**

Any open issues that you aren’t sure about, contentious decisions that you’d like readers to weigh in on, suggested future work, and so on.

#### **Detailed Scoping and Timeline**

Essentially, this is the breakdown of how and when you plan on executing each part of the project.

### **How to write it**

#### **Write as simply as possible**

Don’t try to write like the academic papers you’ve read. They are written to impress journal reviewers. Your doc is written to describe your solution and get feedback from your teammates. You can achieve clarity by using:

* Simple words
* Short sentences
* Bulleted lists and/or numbered lists
* Concrete examples, like “User Alice connects her bank account, then …”

#### **Add lots of charts and diagrams**

Charts can often be useful to compare several potential options, and diagrams are generally easier to parse than text. I’ve had good luck with Google Drawing for creating diagrams.

**Pro Tip:** remember to add a link to the editable version of the diagram under the screenshot, so you can easily update it later when things inevitably change.

#### **Include numbers**

The scale of the problem often determines the solution. To help reviewers get a sense of the state of the world, include real numbers like # of DB rows, # of user errors, latency — and how these scale with usage. Remember your Big-O notations?

#### **Try to be funny**

A spec is not an academic paper. Also, people like reading funny things, so this is a good way to keep the reader engaged. Don’t overdo this to the point of taking away from the core idea though.

#### **Do the Vacation Test**

If you go on a long vacation now with no internet access, can someone on your team read the doc and implement it as you intended?

The main goal of a design doc is not knowledge sharing, but this is a good way to evaluate for clarity so that others can actually give you useful feedback.

### **Process**

Ah yes, the dreaded *P-word*. Design docs help you get feedback before you waste a bunch of time implementing the wrong solution or the solution to the wrong problem. For now, let’s just talk specifically about how to write the design doc and get feedback for it.

At the beginning of this article, we said the goal of a design doc is to **make sure the right work gets done.**