Ingesting Discharge List

A rural hospital and their employed PCP group needed to improve their transitional care management process so that patients get the support they need after they get discharged from the hospital in the comfort of their home. However, due the technical barriers in interoperability, critical information — who got discharged, when and for what reason — is not always available to the PCP group in a timely manner (within couple days). Hospital EHR is not advanced, and paper is the main medium to disseminate information — meaning you can't just send a `GET patients/discharge` request to integrate that into Kouper platform. However, we are in luck and the hospital case manager kindly agreed to send us a discharge list in **PDF format** which can be exported from the hospital EHR. We will need to ingest this discharge list so that we can trigger outreach campaigns for them.

Q1 : How would you approach the problem to go from an unstructured text (pdf) to a structured output (json/xml)?

Draft a technical spec to explain the design space for architecture and relative merits of each approach.

Q2 : Implement a design you described above with a framework of your choice.

You are implementing API endpoint and corresponding UI elements (minus or plus auth) with a framework you are comfortable coding in. A boring (popular and mature) setup is Express + React + RDBMS, but we prefer working code more than the stack.

- Each discharge will require a manual review and we will need to support this **review** process and capture the data (who approved and when).

- Errors in healthcare are not uncommon. For example, if we find out that the name is spelled incorrectly, how do we allow the coordinator staff to edit the name and continue with the campaign while capturing the data lineage ?
- Each data elements can be **decorated** and **enriched** by an external service, and that enriched version is critical for the review process. For example, we can check the phone numbers via an external API to see if it's mobile or landline or even a valid number. https://www.twilio.com/docs/lookup/v1-api/validation-and-formatting Same goes for the insurance information and provider information.

Q3: How would this approach scale as Kouper grows ?

The next customer may have a large volume (1000s of discharges per day) or they may have more advanced EHR which you can send API request to, or it may allow you read into their database, or they may agree to send you HL7 messages. How would your design captures this "future state" reasonably enough while solving the immediate pain ? Think about the any dimensions: variable discharge volume, additional partners who maybe interested in the discharge information, or vendors we might want to integrate into the process.

Things to remember

- A **running code** is more important than non-functioning good code.
- Conditioned on the previous point, **small things matter**. How you name the variables/functions, and other details are just as important as how you structure your program. God is in the details — Ludwig Mies van der Rohe
- **Less is more**: When you approach the problem, try to do more with less. For example, a concise yet complete tech spec (ask in Q1) is a good reflection of your thought process — Watson and Crick pretty much won Nobel Prize with those pages
- **Be ready:** to think and answer any other questions that might pop up.