Mobile Architecture Challenge

Thanks for taking the time to interview for a position at League! To assess your architecture skills, we've designed an exercise to be completed on your own and then reviewed during a remote interview with League. Take as much time as you need to come up with a solutions, but League's expectations are that candidates will spend at most around one hour preparing a solution

Please also write 4-5 sentences explaining how you arrived at your solution. Your description should be concise and speak to the work performed. This write up will be assessed as part of the overall architecture challenge

Architecture Challenge

One of the most important features in the League platform is the ability to manage uploads of documents and claims; documents and claims are images with accompanying metadata. These uploads may exist in a few different places:

- **Claims**: Images supporting a claim, like receipts, prescriptions, doctor's note, etc...
- **Documents**: Personal or recurring documents, like your ID, gym membership, Beneficiary Form, etc. in image format
- Our Customer Care team needs to be able to review these documents for accuracy in their own section of the app (the architecture for this customer care tool is out of scope)

Let's do a high level front-end architecture of what a new upload management system might look like to a League member. There's a few actions to keep in mind:

- The member would like to upload items
- The member would like to view uploaded items
- The member may want to edit an item later
- Submitting a claim is in a different spot in the app than supporting documents; the UX will be similar for both claims and documents, but claims has different metadata than documents:
    - Claims metadata: dollar amount requesting for the claim, Additional notes
    - Documents metadata: Type of document, Additional notes

Here's an example of what the views might look like, feel free to deviate from it, but make sure you support the requirements listed above:

1. Documents/Claims List: A view that contains a list of documents or claims, tapping on a document/claim opens a detailed view. Additionally there's an `add` button that allows the user to submit a new document/claim.
2. Document/Claim Details: A view that shows all of the details for the claims, this includes the image/PDF and any meta-data associated with it . The user can edit and resubmit a document/claim from this view
3. Add document/claim: A view that allows the user to pick or capture an image, add required metadata to the claim and save it.

Deliverables

1. *Low fidelity UX wireframes of the views, showing the UI components that each view will contain. Don't spend a lot of time on this, we just need a basic UX that we can reference for the rest of this challenge, the focus of this challenge is on software architecture, not UX design*
2. **A solution diagram**, produced in draw.io, that shows system components and how they interact with each other. System components can include things such as classes, databases, API clients, etc. We want to understand what the flow of information will look like between the UI and the API. The scope is limited to the front end, you can assume the backend is a blackbox REST API that has whatever API calls you will need to implement your solution.
    - *Make this as detailed as time permits, it should be the main focus of effort and time spent in this challenge*
3. A high level data model of any data that is processed by the system
4. A list of assumptions made for the solution
5. A list of proposed changes and future considerations the solution
6. The above can be stored in a single Google doc and shared with your reviewers at least 24 hours before your interview

Considerations to discuss during your interview

- What architectural patterns were used and why?
- How are errors handled?
- How will it be tested?
- How can the system be changed to handle varying requirements?
- How can the solution be reusable via the slightly different claim and documents UX?