Lingo API/Backend requirements by feature

Data types

Where marked with a *, it would probably make sense to fill a certain requirement with a reference to another piece of data – for instance, returning a patientId rather than a patient.

Tag

A tag must carry the following information:

1) The description written by the practitioner that took the photo / video.
2) The time that the media was captured.
3) The patient(s) that were tagged with this media*. (if this is null, as there is no patient, the practitioner & designation fields can also be empty)
4) A reference to the practitioner that took the photo / video, and the designation they selected*.

Patient

A patient must carry the following information:

1) Date of birth
2) URN
3) Full name

Practitioner

A practitioner must carry the following information:

1) A reference to their associated MXUser*.
2) Their official name (although, this could be stored in an MXUser's displayname)
3) Their job title* (this could link to roles or something).
4) Their email address (this exists on an MXUser, but is private).
5) Their phone number (likewise, this exists on an MXUser, but is private).

Service

A service carries the following information:

1) A location
2) A phone number
3) A name

Role

A role carries the following information:

1) A display name (eg: ED Acute SRMO)
2) A longer name (eg: Senior Resident Medical Officer)
3) A longer description (eg: Emergency Department Acute Senior Resident Medical Officer)
4) An associated designation*. (I'm not certain about this one, as I don't have a perfect understanding of the differences between a role and a designation)
5) A Category (eg: Emergency) *. This should probably be a reference to a category object, so that queries can be performed for all roles of a given category.
6) A Location (eg: CH {Canberra Hospital}) *. Again, this should probably be a reference to a location out of a set of locations.
7) An Organization Unit (eg: ED {Emergency Department}) *. This should probably also be a reference to an organization unit in a set of organization units.

8) A unique role ID which can be used to perform queries using the role.
9) A matrix user ID, so that a chat may be initiated with the role. This can be the same as the unique role ID, but it doesn't need to be.

Call

1) A call type (eg: Lingo Audio, Lingo Video, Lingo Conference, External) – this should probably be represented by an enum
2) A call direction (eg: Outgoing / Incoming / Missed / Not picked up)
3) A call date / time (represented by a timestamp)
4) A call length (represented by an integer count of the number or seconds, or something similar). This could also be the end-date / time of the call.
5) The display name of the person the call was with
    a. If the call was made within Lingo, this could be a user ID, so the matrix user could be queried, and the calls screen could allow the person to be contacted through a chat.
    b. If the call was made outside of Lingo, this should be a phone number, or if the phone number is known, a readable name for the other party of the call.
6) Whether or not the other party involved in the call is known / trusted.
7) An optional phone number, if the call was external.
8) If the call was internal, an Extra Call Info object

Extra Call Info

1) A payload
2) An extra info type (eg: PickedUpBy – to indicate which practitioner picked up a call made to a larger group (eg, a role), or Destination – to indicate the destination of a failed call)

Media Gallery

The media gallery as has been discussed will be implemented as a room storing a specific practitioner's captured media. As such the backend requirement here is simply to create a room for each practitioner and populate the message history with their media.

Locating this "Media Room"

Resultantly, the clients will need a way to determine which room to look in for media. This could be achieved in two ways:

1) A "get practitioner media room" API
2) A fixed naming/ID scheme for the system generated media rooms – for instance, room IDs could be `{matrix-user-id}media`.

Media Tagging

Currently, in the iOS version, patient tags are mock implemented by an API that sends the image URL and receives tag data. This tag data includes the associated metadata and tag history for the tagged media. An API to return this information should return an array of *Tag* objects.

There needs to also exist an API that, when given an image URL, and a set of new tag data, will update or create the tag. This API will be called after the practitioner finishes tagging a piece of media immediately after upload time (as the URL of an image is returned to the client after the image is uploaded).

Directory

Roles

1) There needs to be an API that takes a practitioner (via their matrix ID) and returns a list of role IDs that they occupy.
2) There needs to be an API that returns a role, when given a role ID.
3) There needs to be an API that allows a text-based query to be performed on the roles directory.
4) There needs to be an API that will return all extant roles. This can be the same as the Query API, and just work on an empty query string.
5) An API needs to exist to list all unfilled roles.

Practitioners

1) There needs to be an API that takes a role, and returns a list of practitioners that are filling it (by ID)
2) An API needs to exist that will return domain specific data about a practitioner, given their Matrix ID.
3) There needs to be a text-based practitioner query API (although, this could be achieved through the existing directory APIs used by matrix).
4) There needs to be an API that returns all practitioners (to fill the practitioners/people directory)

Services

1) An API needs to exist that will return a list of services.
2) An API needs to exist that will perform a text-based query on the list of services.

Favourites

1) An API needs to exist for all directory categories that lists the logged-in practitioner's favourited items
2) An API needs to exist for all directory categories that can update the favourite status of a directory item for the logged-in practitioner

Calls

1) An API needs to exist that will list all recent calls made by and to the logged-in practitioner through Lingo.

Server behaviour

Creation of groups / rooms

When a chat is created with a role, all current members of the role need to have the chat content mirrored to them, and their replies need to be injected into the room (from the perspective of the creating user). The rooms should legitimately contain the people they're supposed to contain, so existing features (such as the room member list) continue to work as expected.

Media

All media needs to be replicated into the user's media room.

Calls

Calls to a role need to be directed to the members of a role.

Some method needs to be made such that an external call can be made. This could be by creating a room (or selecting an existing room), then sending the phone number to the room, then initiating a Lingo voice call within the room, or by an API that allows the same kind of functionality.