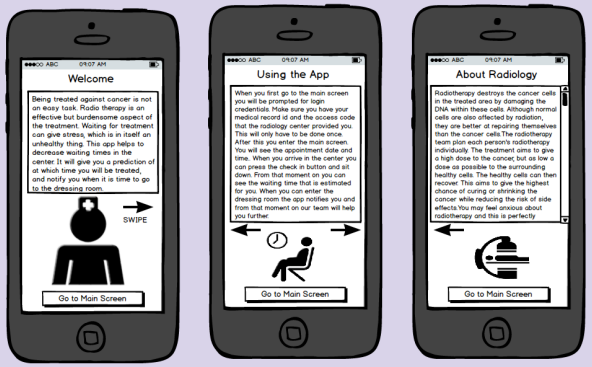
PATIENT APP LAYOUT

The patient app is meant to help reduce waiting times and forthcoming stress from that. Patients register once with their hospital id and a code they got from the radiology center. After that patients are logged in automatically and have access to some of the resources of the Impatient Service. At the appointed date they can check in when they arrive at the center. The app informs them about the predicted waiting time.

Startup of the app

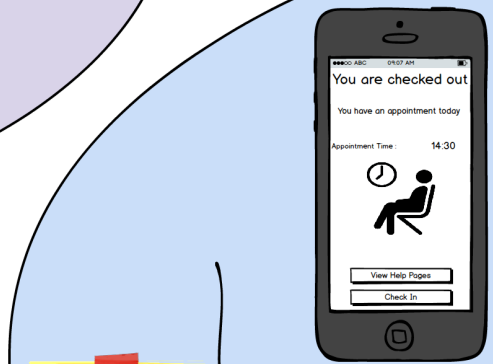


Help pages that help navigating the app and giving some backgrounds of the treatment.; The helppages are formatted as a ViewPager.

Press on Main Screen Button

Succesful Login

Subsequent openings



Main screens. The screens vary dependent on appointment date and checked-in status. A change of status will be reflected in dependant applications.

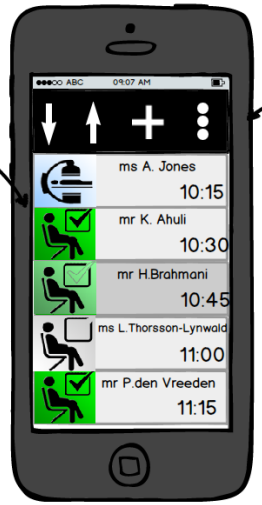
A patient gets a notification with a sound an a message in the notification bar when he is first in the queue to receive treatment.

ADMIN APP SCREEN LAYOUTS AND ACTION FLOW

You see a list of people that are in a queue to undergo radiation treatment. The flow of this procedure is not transparant: People are coming and going, people are delayed, admins can't control this procedure any more. This is a source of stress for patients and staff. The app illuminates which people are presen, available , still have to come, are under treatment. Patients get their estimated waiting time, so that they can take a stroll or a refresh. Admins can insert unforseen patients or move patients up the list when a machine is bound to run idle.

Click the coloured icon to change the status :
Available -->after a Patient checks in or when admin checks in manually
Currently Unavailable --> Patient is checked in but cannot be called immediately
Unavailable --> No check in via the app or via the staff
Under Treatment -->Patient will be notified

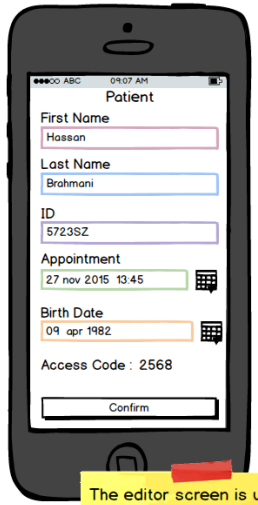
A swipe gesture removes a patient from the list



Action bar items :
4 choices :
Move selected item up
Move selected item down
Insert a patient in the list above the selected patient
More Options :
registering a new patient
making a new appointment
going to the patients list
going to the help pages
changing the settings

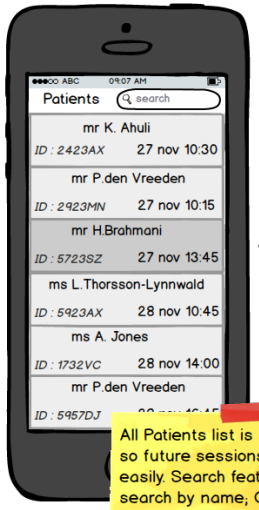
Registering a new patient

Making a new appointment



The editor screen is used to make new appointments, change patient data , or to view a users access code for the app.

Going to the patients list

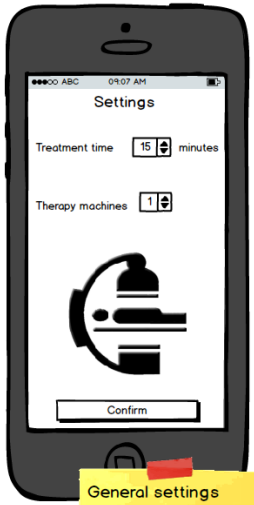


Item click

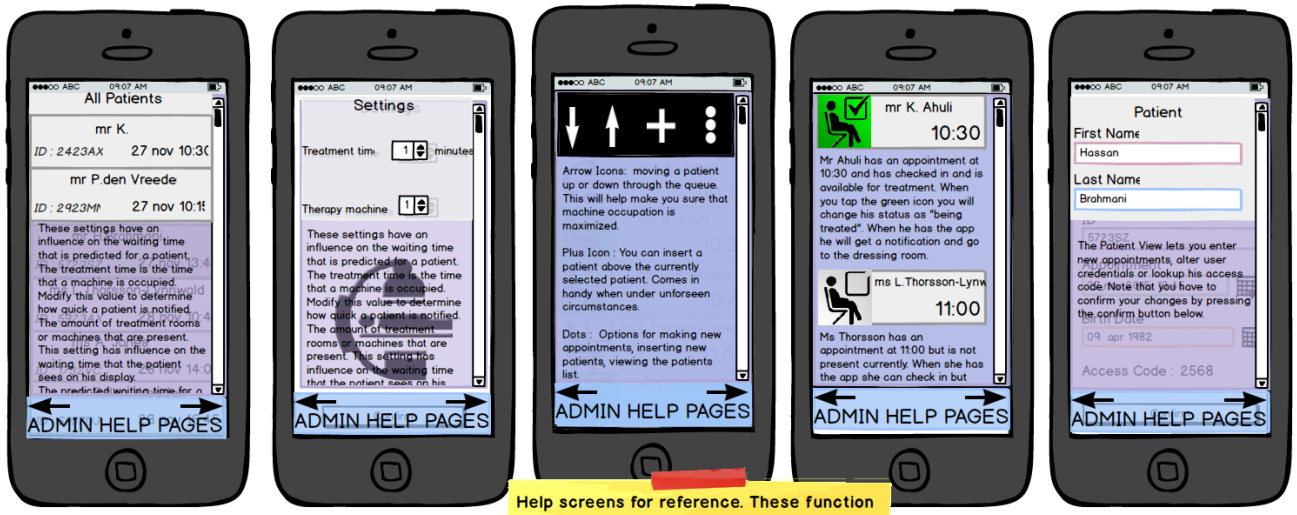
All Patients list is sorted by date so future sessions are spotted easily. Search feature added to search by name; Clicking an individual item goes to the Patient screen where new appointments can be made.

Changing the settings

Main entry point. Every time the app restarts credentials are being requested. This is different from the user app. However all patient data are visible here so care must be taken.



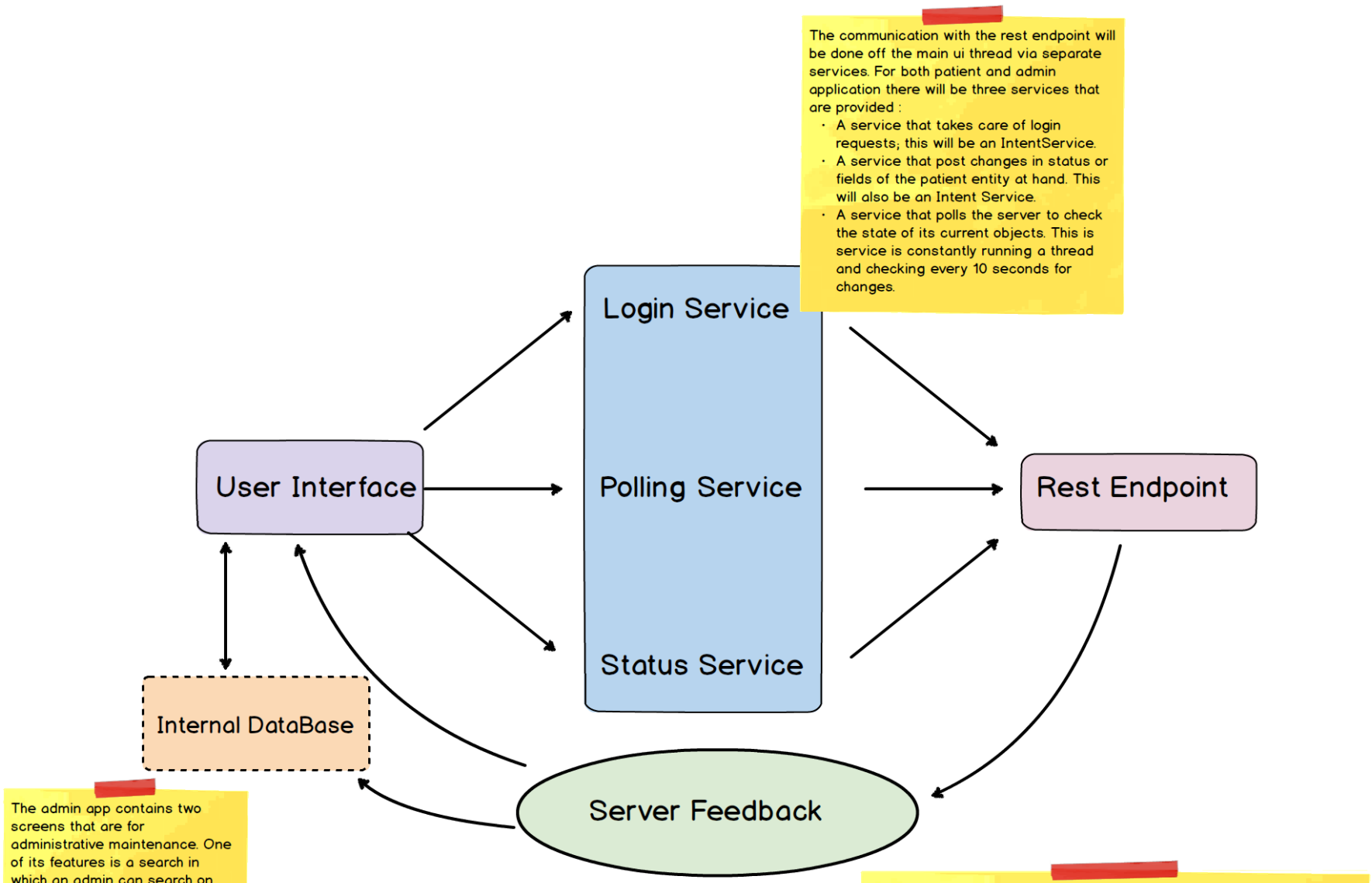
General settings that are used to predict waiting times.



Help screens for reference. These function as a ViewPager booklet through which you can swipe quickly through the pages.

Android Clients structure

Patient app and Admin app



The communication with the rest endpoint will be done off the main ui thread via separate services. For both patient and admin application there will be three services that are provided :

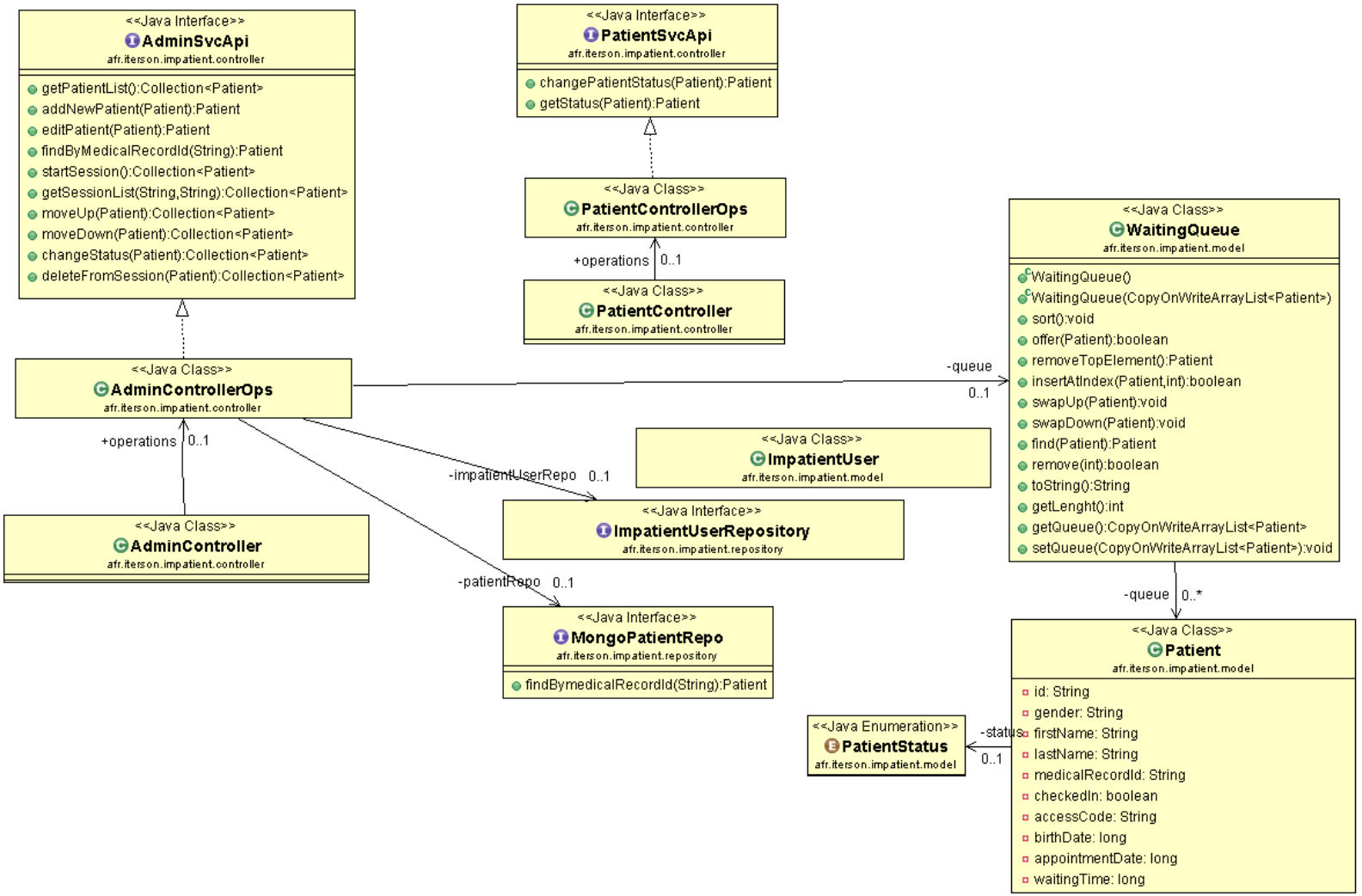
- A service that takes care of login requests; this will be an IntentService.
- A service that post changes in status or fields of the patient entity at hand. This will also be an Intent Service.
- A service that polls the server to check the state of its current objects. This is service is constantly running a thread and checking every 10 seconds for changes.

The admin app contains two screens that are for administrative maintenance. One of its features is a search in which an admin can search on names, dates and times. A content provider would facilitate this very easily, and save unnecessary calls to the Rest Endpoint. The patient app however will have no need for this feature.

Most calls from the 3 services will get replies back from the server. A typical flow of events is as follows :
1. patient app makes a request for the current waiting time.
2. Using the medical record id the server calculates the waiting time for this patient.
3. The service receives a patient object back.
4. A broadcast will be made to a listening BroadcastReceiver
5. In its onReceive() method the BroadcastReceiver will update fields in the user interface or trigger an alarm when the waiting time happens to be zero.

The admin app will follow this flow. Only now we are handling a list of patients that we are going to update.

Basic class layout Spring server part



The server part follows roughly the division between the two client apps. Two apis define the methods and the paths that this server part will use. These apis will also be used by the clients so that we are sure that all methods and paths exist.

The controllers are the actual beginning point for https requests but let the actual work be done by the controllerOps objects. Here is also the scope defined that allows permission for a particular method.

The main object that is used is a WaitingQueue that functions as an application wide singleton. It carries a CopyOnWriteArrayList that ensures thread safety. We choose an ArrayList because it maintains its order, an essential thing when you're working with queues.

The Patient object is a straightforward Java pojo that contains the fields we want to monitor throughout the application. A Collection or an individual Patient will often be used as the return type for a client request.

The MongoPatientRepository stores the Patient objects and is backed by a Mongo database. The choice for a non relational Mongo type is because the object one uses almost map perfectly towards the rows/documents.

The ImpatientUserRepository carries the user credentials for all users of the app. Administators can register new users or change credentials in this repository. The Oauth2 security configuration constructs its User Details Service out of ImpatientUser objects, that in their turn are pulled out of the database.

Implementation Considerations

Following the Coursera "project requirements" section

How will a Patient enter identifying information upon Check In? In particular, what will the user interface look like for a Patient so that it is quick and simple to use?

After first opening the patient is asked for its hospitalpatientid and a unique code the patient received after registering in the hospital. By asking the unique code we make sure that no one who might have access to her hospitalpatientid can access the app in her place. It is assumed that the combination of the two warrants a user that is authentic. After that we don't ask credentials any more. Subsequent starts will immediately go to a check in screen and connect with the server.
The user interface will contain of a single screen with three buttons : check in, delay, and check out. The appointment time and date is displayed

How will Check-In data be transferred from a Patient's device to the Admin? Will this be done via the app or not?

When a patient checks in there will firstly be made a call to the server that checks whether an appointment exists at this day for this particular patient.
If not we will ask for confirmation. If the patient insists the patient will give in the presumed appointment time after which he will be added to the waiting queue at the appropriate place, even when there was no appointment. The user will be informed to contact to staff after arrival. The server transmits the list to the admin.
When there is a regular appointment the patient is added to the list, and all the open admin apps are notified that this particalar patient is ready for treatment.

How will the app handle concurrency issues, such as how will periodic updates occur - via server push or app pull? How will queries and results be efficiently processed? Will the data be pulled from the server in multiple requests or all at one time?

Each app will have multiple services running to make https requests:
An admin app will poll the server every 10 seconds to retrieve a listview that contains the latest update of the list. For example, when a patient has checked in, this change is added to the list and visible for the admin after a refresh. This service launches a a background thread that does this work. The response of the service will be handled by a BroadcastReceiver.
An admin app will use an Intent Service for the following actions :
Modifying the WaitingQueue.
Changing the treatment times, amount of machines or inserting a lunch break.
Inserting a new appointment.
Inserting a new patient.
A patient app will poll the server every 10 seconds to retrieve the estimated waiting time and to check for a possible notification. This polling works in a continually running background thread and reports via a BroadcastReceiver.
A patient app will use an Intent Service to check in, to delay, or to check out.

How, when, and how often will the user enter their user account information? For example, will the user enter this information each time they run the app? Will they specify the information as part of a preference screen?

Admins have to login for each session. Since there is information that is privacy sensitive they will be prompted again after the app is resumed. The admin names will be available in a drop down menu but the password has to be provided.

What user preferences can the user set? How will the app be informed of changes to these user preferences?

An admin can change the following settings.
the amount of treatment machines that are available. We assume a default of 1 but with more machines the queue velocity increases accordingly. Since patients need to be informed of their estimated waiting time
the average treatment time. The waiting time can be calculated as people before you * treatment time /amount of machines + + shortest remaining time current treatment ; the formula changes, depending on the position in the list.
lunch or tea breaks

How will a Patient's data be securely transferred to the server?
How will you ensure that Patient data is secure and shared only with the Admins?

Transfer will be done via a secure HTTPS connection with OAuth2 authentication.
There will be two different apps, one for admins and one for patients. Patient apps are not able to see lists with other patient information. They can checkin and delay, and be notified by the server when it is their turn.

How will the app use at least one advanced capability or API listed above? For example, will you create an animation to explain the app? Will you allow Patients to take/share pictures of themselves so that an Admin will know who they are and what they look like?

The patient will receive an instruction booklet that works as an animated viewpager which will be shown after first login and later on as an action bar option. Similarly, the admin will have the admin help pages.
In the admin app deletion of a patient goes via a swipe gesture.

Does your app really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider or from using a background Service that synchronizes local and remote data, only when the device is connected to a WiFi network.

Both apps depend on services to function properly. Local data storage will be done sparingly, except for Patient Credentials and notification preferences. A content provider will be used in the admin app to facilitate a search function. The apps rely on wifi connectivity to function properly. Both apps will be triggered by broadcastreceivers to update their results in the app.