Website/Server:

Architecture:/How is the code organized into modules:

The source code for the server is divided into modules that essentially follow the traditional Django framework pattern. In the ultraRemoteMedicine subdirectory, __int__.py can be used to add any custom start-up scripts that might be needed. settings.py and wsgi.py define various parameters and settings for Django. When a user sends a GET or POST request to the server, urls.py interprets the url to determine which *view* it corresponds to.

A view is essentially a method that formulates a response to that request. The views are located across various python files in the sample directory, such as case.py, doctor.py and patient.py. The views normally return html code, with the template for that code coming from files in the templates/website directory. Most of the forms used for communicating with the views are found in forms.py rather than being hardcoded html. Django abstracts away most of the work in dealing with the database, which is represented in models.py.

How it interfaces with existing libraries support software :

We use Django for the framework for our website since it is easy to use and comes with a lot of built in functionality that makes our lives easier, and we can focus on the important tasks. It also comes with an admin page. We are using the Heroku cloud computing platform to host the website since we could not get Windows Azure to work, and Heroku is an easy-to-use and easy-to-learn platform. For our database, we use postgreSQL since it is compatible with heroku. We use HTML5, CSS and Javascript for basic web programming, as well as the twitter bootstrap package for making the webpage stylish. The all the pages were made with twitter bootstrap.

Known bugs:

Images uploaded on the web may be deleted after few hours.

Features included:

- Administrator
  - add/delete doctor's account
    - Create basic information for doctor, such as account username, password, specialties, available schedule and personal comment
    - delete the expired account if require
  - add/delete fieldworker's account
    - Create basic information for fieldworker, such as account username, password

- Doctor
  - check existing patient's profile which includes patient's case and basic information.
  - modify the priority for each case and write comments for the patient and

    reply comments to the submitters along with scan images.
  - modify personal information
- Fieldworker
  - check existing patient profile which includes patient’s case and basic information.
  - uploading scans.
  - modify the priority for each case and write comments for the patient and reply comments to the submitters along with scan images.
  - modify personal information
  - create profile for new patient
  - add new case for patients.

<u>Optional features:</u>

- sortable table : make each column of the table ascending or descending order
- a search bar for doctor or fieldworker to search the information he/she needs by entering the keywords.

For Android ->

ActiveAPI.java is the module where it uses the JSON api to connect to the database to withdraw informaiton. This module is used by other modules to get the information from the database.

And it every other .java file in the src folder is a separate screen which uses ActiveAPI.java to gather information and display it.

In the res/layout folder there are xml files to display the layout. It uses .java files to display objects.

Features

Log in and Log out of the app.

Search a new patient

Add a New patient and fill out a form.

Add a Case to the patient with an option to add a photo and comment.

JARICK
Add an acknowledgments section to this (I think), with these plugins we used:

Acknowledgements:
jquery: http://jquery.com/
Datatables: http://datatables.net/
jquery file upload: https://github.com/sigurdga/django-jquery-file-upload
Search: http://julienphalip.com/post/2825034077/adding-search-to-a-django-site-in-a-snap