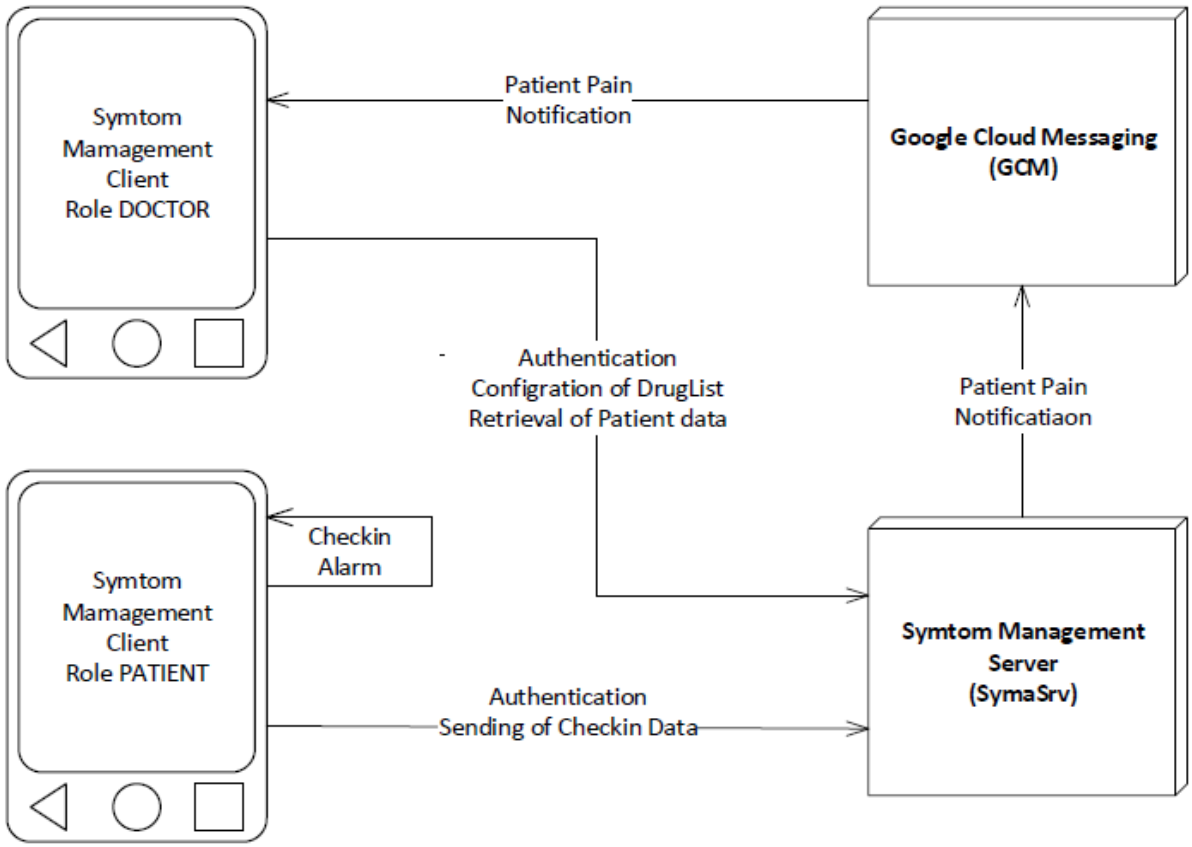
Android Capstone Mid-point specification

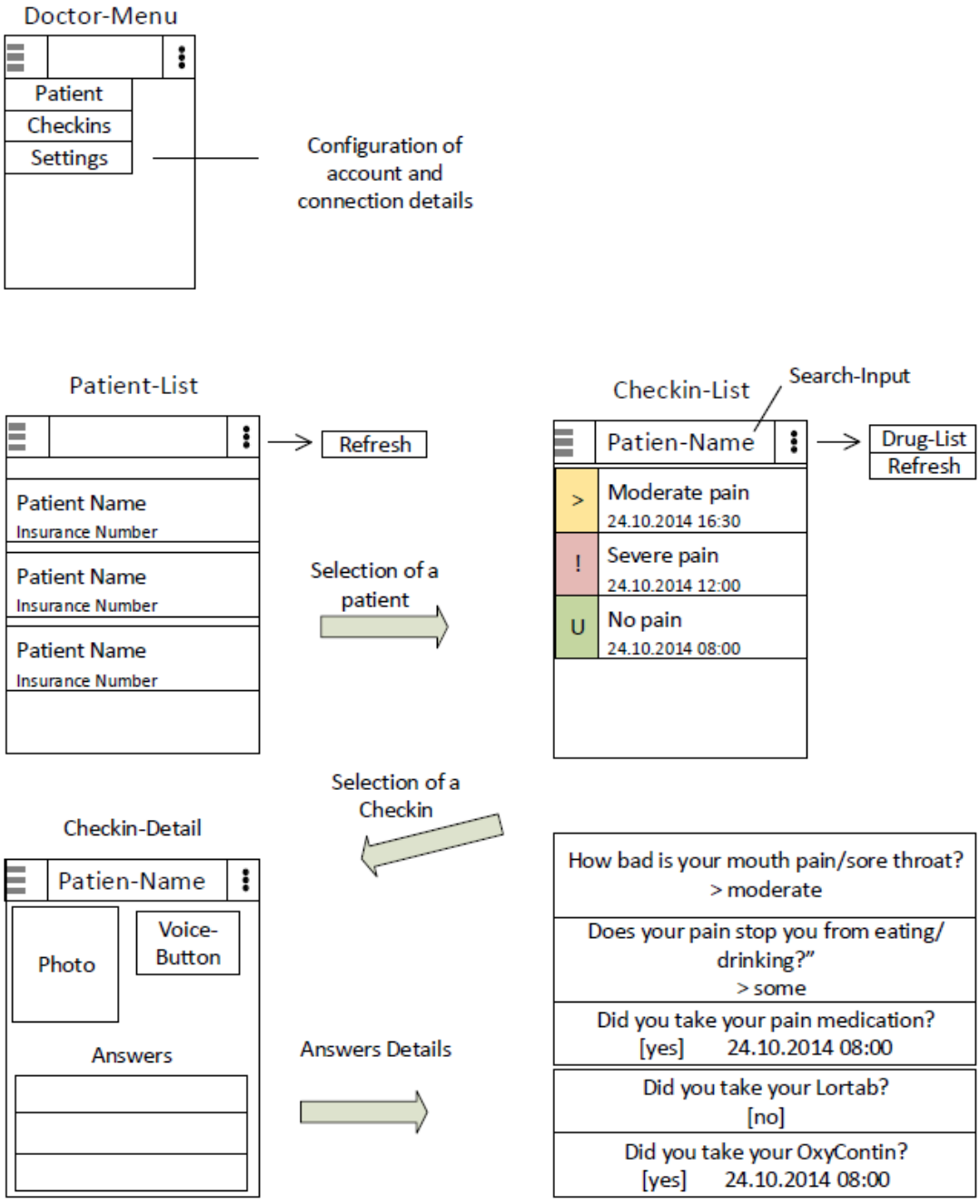
Option C: Symptom Management

Architecture Overview and Deployment Diagram

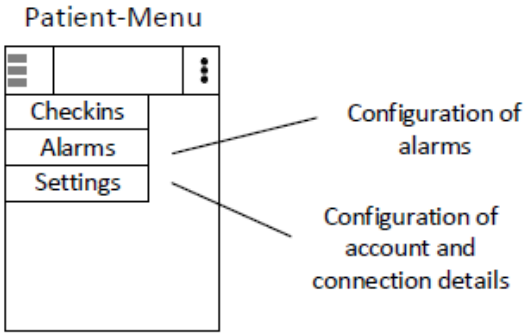


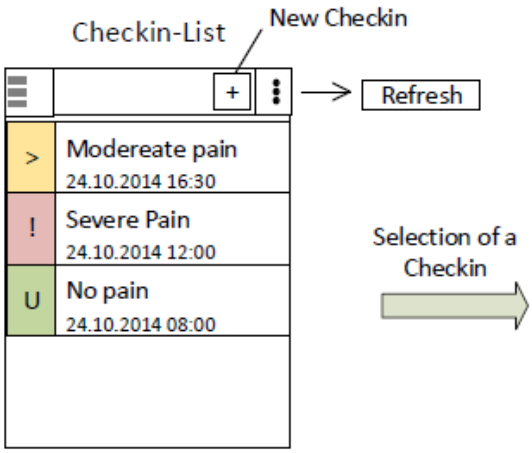
- The Symptom Management application consist of a server and a client part.
- The server is implemented using Spring Boot
- The client is an Android application.
- Depending on the users role the application will run either in Role DOCTOR or in Role PATIENT modus.
- The communication between client and server is using JSON messages of HTTPS.

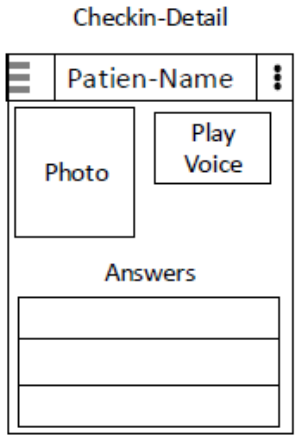
User Interface for role DOCTOR

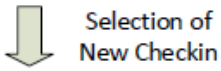


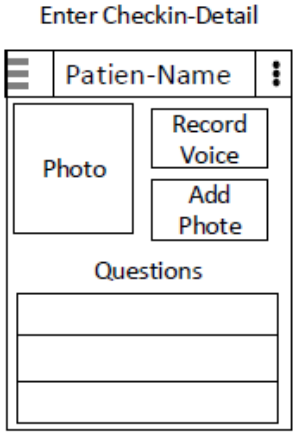
User Interface for role PATIENT

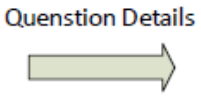








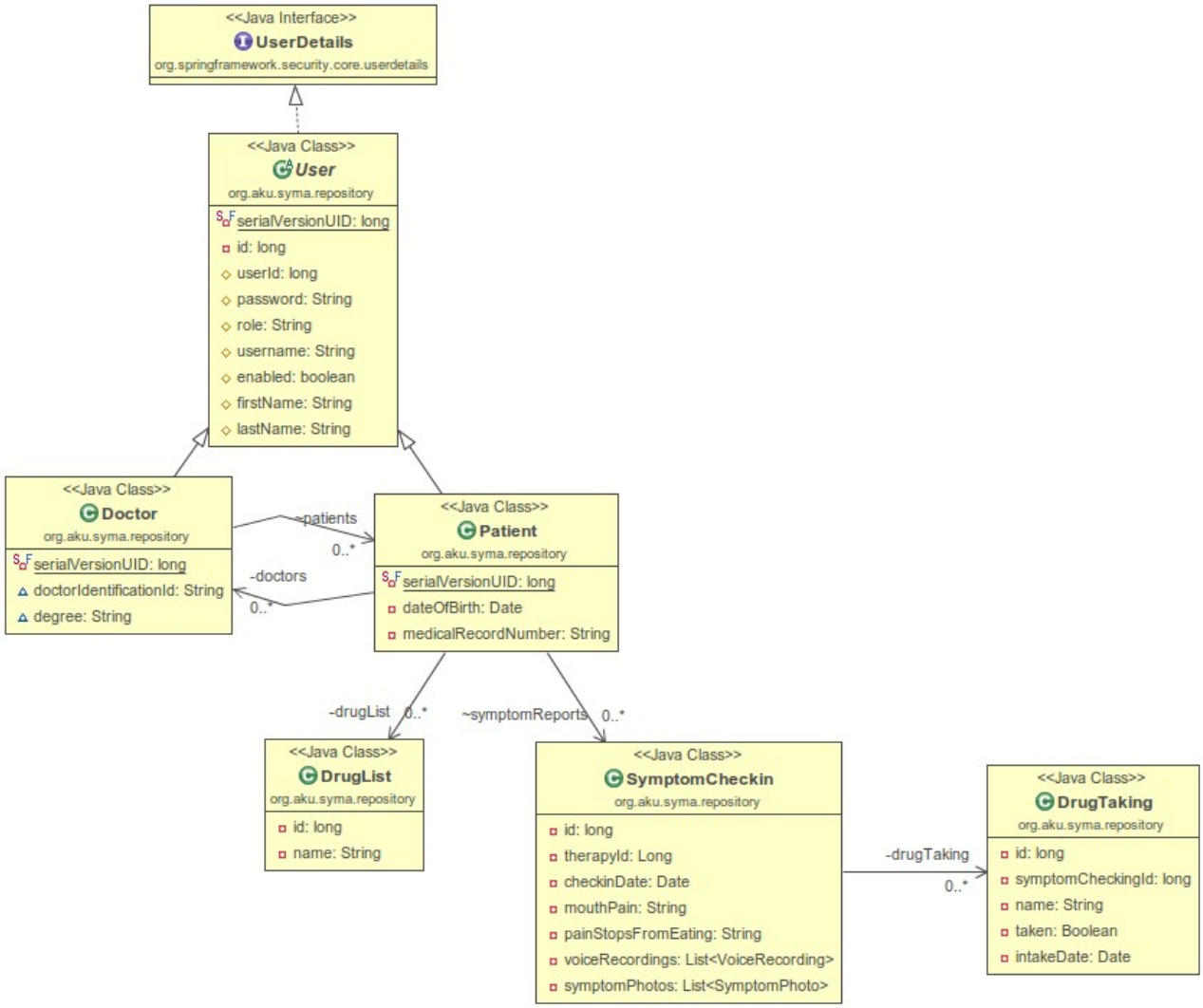




Three Generic questions allways asked.
Additional questions depending on DrugList.

How bad is your mouth pain/sore throat? U well-controlled > moderate ! severe
Does your pain stop you from eating/ drinking?" U no > some ! can't eat
Did you take your pain medication? [yes] [no] [Date-Time-Selection]
Did you take your Lortab? [yes] [no] [Date-Time-Selection]
Did you take your OxyContin? [yes] [no] [Date-Time-Selection]

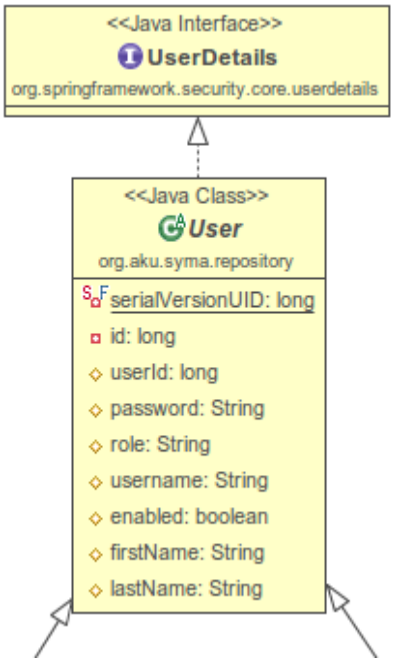
Class Diagramm of entities



Basic Project Requirements

1. App supports multiple users via individual user accounts

A generic User class will be holding user account data. A user, which is either a doctor or patient has to login to the system to get access to it. Authentication is performed using HTTP basic authentication. Account data will be stored in the entity User. The entity User is implementing the Spring UserDetails interface so that is can be utilized by a Spring UserDetailsService. Each user has a security role assigned which is either ROLE_DOCTOR or ROLE_PATIENT.



2. App contains at least one user facing function available only to authenticated users

Any access to the system will be authenticated by configuration of a WebSecurityConfigurerAdapter.

Example of role based authentication:

```
http.authorizeRequests().antMatchers("/doctor").hasRole("DOCTOR");
http.authorizeRequests().antMatchers("/patient").hasAnyRole("DOCTOR", "PATIENT");
http.authorizeRequests().antMatchers(HttpMethod.POST, "/checkin").hasRole("PATIENT");
http.authorizeRequests().antMatchers(HttpMethod.GET, "/therapy").hasAnyRole("DOCTOR", "PATIENT");
```

Example: A doctor may see the checking of a patient but he cannot add a new one or modify an existing one.

3. App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components

3.1 Activity

The main screen will be implemented as an Activity with an action bar. Further screens will be implemented using Fragments.

3.2 BroadcastReceiver

The doctor instance of the application will have a broadcast receiver instance to retrieve alarms according to the rules for patient checkins.

A WakefulBroadcastReceiver will be used to process Google Cloud Messages (GCM).

3.3 Service

IntentService will be used to handle Google Cloud Messages (GCM).

3.4 ContentProvider

Access to the sympttom checkin data will be performed by the means of an Android ContentProvider.

4. App interacts with at least one remotely-hosted Java Spring-based service

All the data for the application will be managed by a Java Spring-based application server. Access will be provided by the means of Spring-based services (JSON over HTTP).
There will be at least to services
a) UserService – Handling user login and Doctor / Patient data.
b) TherapyService – Handling symptom checkin

5. App interacts over the network via HTTP

See point 4.)

6. App allows users to navigate between 3 or more user interface screens at runtime

The symptom management app hat two modes: doctor- or patient-mode. The role a user gets will be decided during login.

The user interface screens of a doctor are:

- Patient Overview
- Symptom Checkin Overview
- Symtom Checking Detail

The user interface screens of a patien are:

- Symptom Checkin Overview
- Symtomp Checking Detail
- Symptom Checkin Input'

7. App uses at least one advanced capability or API from the following list (covered in the MoCCA Specialization): multimedia capture, multimedia playback, touch gestures, sensors, animation.**

**Learners are welcome to use ADDITIONAL other advanced capabilities (e.g., BlueTooth, Wifi-Direct networking, push notifications, search), but must also use at least one from the MoCCA list.

The symptom management app makes use of camera and void recording to report the health status.

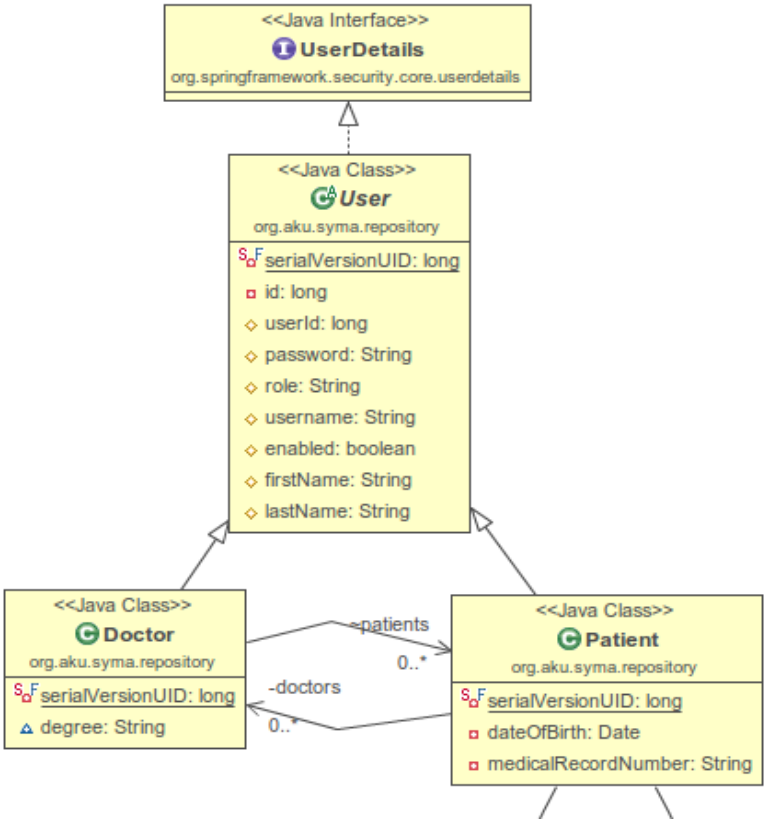
8. App supports at least one operation that is performed off the UI Thread in one or more background Threads of Thread pool.

Data exchange by the Syma Android client with the Syma application server is performed using the Android AsyncTask framework.

Functional Description and App Requirement

1. **App identifies a Patient as a user with first name, last name, date of birth, a (unique) medical record number, and possibly other identifying information). A patient can login to their account.**

Login to the Syma server is realized using HTTP basic authentication and Spring Authentication. Both doctors and patients must login to the Syma server to gain access to the data. There will be three entities to model this relation. A generic User entity used by the login manager and a Doctor and Patient entity to hold doctor and patient specifc data. Doctor and Patient entity are derived from the User entity.



2. **App defines a Reminder as an alarm or notification which can be set to patient-adjustable times (at least four times per day).**

The alarms configuration screen allows the patient to define four daily occurring alarms. There is no GUI-design for the alarm contained in this report.

3. **A Reminder triggers a Check-In, which is defined by the app as a unit of data associated with a Patient, a date, a time, and that patient's responses to various questions at that date and time.**

Please see chapter User Interface for role PATIENT

4. **Check-In includes the question, "How bad is your mouth pain/sore throat?" to which a patient can respond, "well-controlled," "moderate," or "severe.**

Please see chapter User Interface for role PATIENT

5. **Check-In includes the question, "Did you take your pain medication?" to which a**

Patient can respond “yes” or “no”.

Please see chapter User Interface for role PATIENT

6. **A Check-In for a patient taking more than one type of pain medication includes a separate question for each medication (e.g., “Did you take your Lortab?” followed by “Did you take your OxyContin?”). The patient can respond to these questions with “yes” or “no.”**

Please see chapter User Interface for role PATIENT

7. **During a Check-In, if a patient indicates he or she has taken a pain medication, the patient will be prompted to enter the time and date he or she took the specified medicine.**

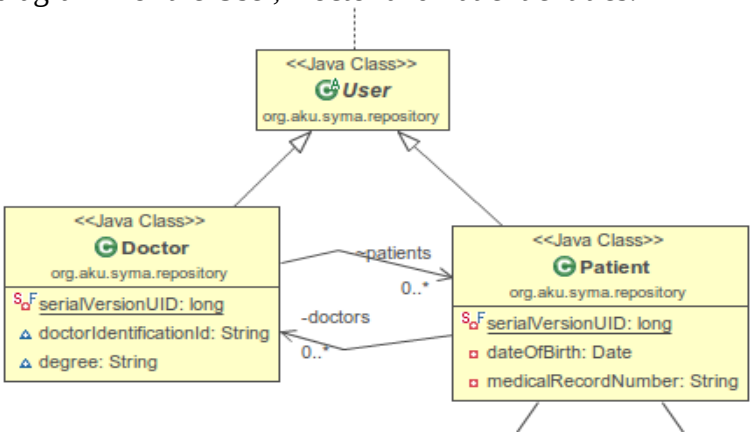
Please see chapter User Interface for role PATIENT

8. **During a Check-In, the patient is asked “Does your pain stop you from eating/drinking?” To this, the patient can respond, “no,” “some,” or “I can’t eat.**

Please see chapter User Interface for role PATIENT

9. **App defines a Doctor as a different type of user with a unit of data including identifying information (at least first name, last name, and a unique doctor ID) and an associated list of Patients that the doctor can view a list of. A doctor can login.**

The class hierarchy diagramm of the User, Doctor and Patient entities.



There is 'm' to 'n' relationship between the Doctor and Patient entities. A doctor treats multiple patients. A patient may have treatments by multiple doctors.

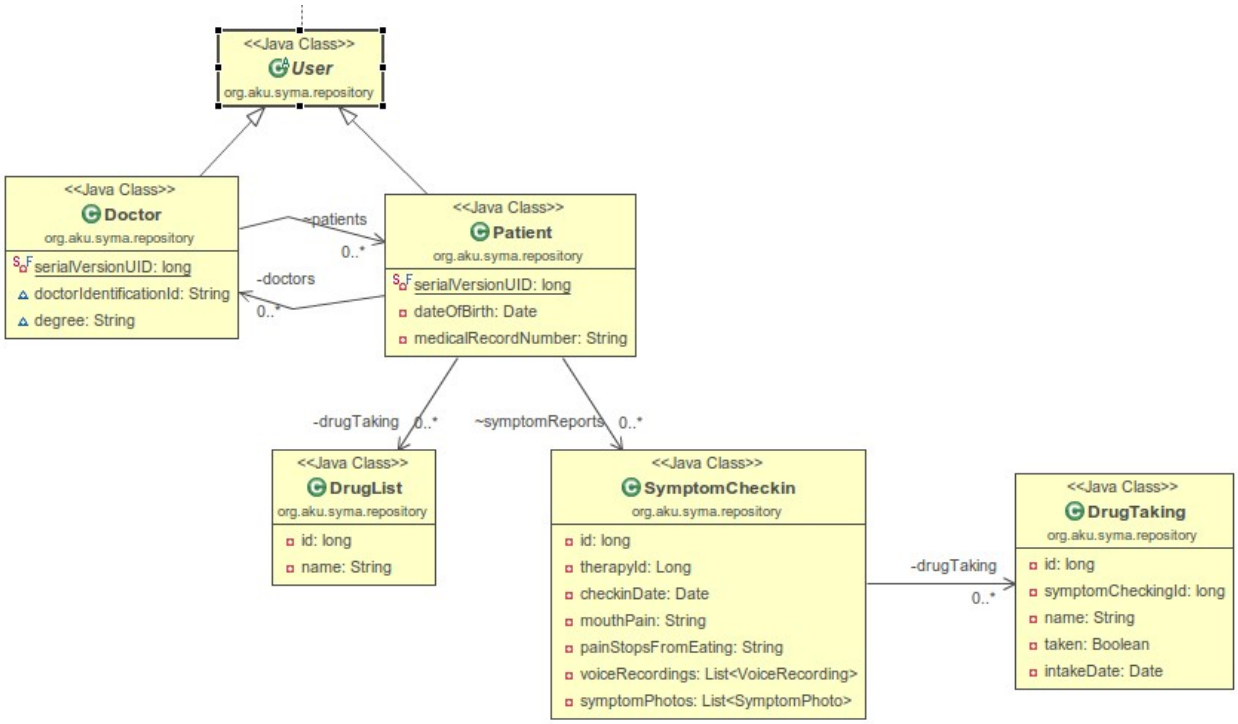
10. **App allows a patient’s Doctor to monitor Check-Ins, with data displayed graphically. The data is updated at some appropriate interval (perhaps when a Check-In is completed).**

Google Cloude Messaging (GCM) will be utilized for communication between patient and doctor application. The doctors checkin list will automatically be updated if the correlating patient is completing a checkin.

11. **A doctor can search for a given Patient’s Check-In data by the patient’s name (an exact text search hosted server-side). Note: Non-exact text searching is not required (e.g. you don’t have to suggest, “Did you mean…”)**

Please see chapter User Interface for role DOCTOR

12. A doctor can update a list of pain medications associated with a Patient. This data updates the tailored questions regarding pain medications listed above in (6).



There is a '1' to 'n' relationship between the Patient and the DrugList.
For each checking there are questions based on the DrugList, The answers will be stored in DrugTakin list. There is a a '1' to 'n' relationship between the SymtomCheckin and the DrugTaking.

The screen to access the Medication List of a patient will be implemted as master detail view. A user may have exactely one medication list associated.

13. A doctor is alerted if a patient experiences 12 of “severe pain,” 16 or more hours of “moderate” or “severe pain” or 12 hours of “I can’t eat.”

The duration of a patients pain will be calculated on the server by a task being periodically activated by the the Spring scheduler. A doctor will be notified using the Google Cloud Messaging framework for Android (GCM). Please see chapter Architecture Overview and Deployment Diagram

14. A patient’s data should only be accessed by his/her doctor over HTTPS.

The symptom management application server will provide authenticated access over HTTPS. Access by HTTP will not be supported.