Android Capstone Project

Symptoms Management

Project requirements

30/11/2014
Coursera: https://class.coursera.org/androidcapstone-001

README First

My solution contains two projects. The first project is located in **SymptomsServer.zip** and contains the **Spring Boot server**. It is an **Eclipse Gradle project** and you can unzip an import it in your Eclipse as a Gradle project following the instructions covered in the third course of this specialization. To start this project you need to "Run as an application" the Application java class that is located at org.coursera.symptomserver package. This class starts a Tomcat server on https port (8443) and remember to add the keystore following the instructions that are present in the Application.java file and covered in the projects example of the third course.

This server uses a Mysql database to store all information. It is needed you first create a database called symptom with a user symptom and password symptom and executes the two scripts that are located in the sql directory. You can change the database name, the user name and password if you want, editing the file application.properties that is located in the src\main\java directory of this project. If you don't have experience with mysql commands the steps for create a Mysql database are the following (Of course, I assume you have a Mysql server instance running in your machine):

1) Create database symptom:

From command line write: mysql -u root -p (intro)

Mysql asks for the root password. Once you enter the password you will be log in Mysql server.

Type: use msyql (intro)

Type: create database symptom; (intro)

2) Create a username and password in msyql:

Type: create user symptom@localhost identified by 'symptom'; (intro)

3) Grant all symptom tables to the new username created:

Type: grant all on symptom.* to 'symptom'@'localhost'; (intro)

4) Create symptom tables:

Now you need to exit mysql with the command: exit (intro)

From the command line go to the SymptomsServer directory where you unzip the project and type: mysql -u symptom -p < sql/Database.sql (intro)

Type the password: symptom

5) Inserts doctors and patients to begin with:

From command line type: mysql -u symptom -p < sql/Inserts.sql (intro)

Type the password: symptom

Both Database.sql and Inserts.sql files assume that the database name is symptom. If you want to change this name in the application.properties file you will need to change the first line of these two files in order to execute these scripts properly.

This project has a Test suite that test all REST operations. You can Run the project as a JUnit and test it. This TestSuite is located at org.coursera.symptomserver.test.ServerTest.java. Change before the constant TEST_URL concordantly with your server installation.

The Test suite creates a lot of checkins and patient medications rows that you'll need to delete before test the Android App for a better result. To do from the command line go to the SymptomsServer directory where you unzip the project and type:

```
mysql -u symptom -p < sql/Deletes.sql
```

The second project is located in **Symptom.zip** and is an **Eclipse android project** that you can unzip and import on Eclipse as an Android project. The target API for this project is 16 as a minimum API and 20 as a maximum API. Before you start an emulator to test the project you will need to change the URL where the server is located. This is a constant with the name **SERVER** that you can find at org.coursera.symptom.SymptomValues java class.

To test the android project you can be identified as a patient with the username: patient1 and password: patient1 and as a doctor with the username: doctor1 and password: doctor1

It is important to know that it is better to use a different emulators, one for patient and another for doctor. If you login in the same android emulator with both users, every time you change the user the app will delete all previous information inserted by the last user in the local database but not, of course, in the server database. This is done for security reasons to prevent that one authenticated user can view information in the mobile app that belongs to another user. For example if the user steals the mobile.

Basic Project requirement

1. App supports multiple users via individual accounts

At SymptomServer spring project you'll find the OAUTH configuration in the class org.coursera.symptomserver.oauth.OAuth2SecurityConfiguration that supports multiple accounts. I assume for this project, in memory authentication following the example that we saw in the third course of this specialization. The OAuth2config() constructor at line 123 in the inner class OAuth2Config configures what users with what roles can access to the system. The Main Java class org.coursera.symptomserver.Application import this oauth configuration with the annotation:@Import(OAuth2SecurityConfiguration.class) at line 51.

At Symptom android project, you'll find in the org.coursera.symptom.activity.LoginActivity class the login() method at line 72 that allow clients to authenticate in the server with their individual accounts. When a client insert their username and password the android app calls /symptom/getstatus method that identified who is the user. You can find this method at SymptomServer Spring project at org.coursera.symptomserver.controllers.SymptomController class at getStatus() method at line 62.

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

In my solution all functions are only for authenticated users. In fact, all REST API is protected by DOCTOR or PATIENT Role and only users with a valid account can authenticated with the system and works with the App. The inner class ResourceServer inside the org.coursera.symptomserver.oauth.OAuth2SecurityConfiguration class makes this protection at line 78, method configure().

3. App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components: Activity, BroadcastReceiver, Service, ContentProvider

My solution uses all of this fundamental components. In Symptom Android project you can find **Activity** classes in the packages org.coursera.symptom.activity. Two **BroadcastReceiver** in the package org.coursera.symptom.receivers. These BroadcastReceiver are used to receive a broadcast sent by an alarm that is used to send a reminders for patients and another one is used to receive a broadcast send by Android when the device is reboot to set the alarms again. Two **services** in the package org.coursera.symptom.services, one for send the patient's checkin in background and the photo that the patient can take and another to download checkins, patient medications or patient's photos. Finally, a **ContentProvider** to access the local database to reduce traffic between app and server. This is located in the class org.coursera.symptom.provider.SymptomProvider. All of these components are defined in the AndroidManifest.xml file.

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

My App interacts with an API Rest that you can find in the SymptomServer spring project at class org.coursera.symptomserver.controllers.SymptomController. This REST API is published at URL https://localhost:8443/symptom and you can find an example of interaction in the App in the Symptom Android project at class org.coursera.symptom.activity.doctor.PatientListActivity in the method showPatientList() at line 90 that download the doctor's patient list from server. But there are a lot of other examples along the code. For example at class org.coursera.symptom.activity.doctor.MedicationsFragment at lines 178, 215 and 224. At class org.coursera.symptom.services.SendIntentService at lines 79 and 90. And at class org.coursera.symptom.services.DownloadIntentService at lines 199, 223 and 254.

5. App interacts over the network via HTTP

My App interacts via HTTP to access the server using retrofit library that we saw in the third course. You can find this at Symptom Android project at org.coursera.symptom.client.SymptomSvc class at line 61. This method builds a retrofit object of org.coursera.symptom.client.SymptomSvcApi class. When this retrofit object is built an EasyHttpClient is set to make the communication over http. This is done at line 67.

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

At Symptom Android project, you can follow this sequence: org.coursera.symptom.activity.LoginActivity class that presents a login screen, then this activity calls the org.coursera.symptom.activity.doctor.PatientListActivity activity at line 148. This activity shows a patient list. From patient List when the user presses a row, the activity calls org.coursera.symptom.activity.doctor.CheckinListActivity at line 69, method startCheckinListActivity(). This is a method that you can find at org.coursera.symptom.activity.doctor.BaseActivity class. The CheckinListActivity class shows a patient's checkin list. When the user presses a row from this list the activity calls org.coursera.symptom.activity.doctor.CheckinDetailActivity at line 51. This activity shows the checkin information. And finally from this activity the user can presses a button to see the patient photo that took at the end of the checkin process. This call is done at line 261, method showPhoto().

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

There are two advanced capabilities in my project. One that uses a camera to allow patient to take a photo to send to doctor's app. This can be useful for patients to show their mouths/throat to their doctor, maybe to control blisters or sores. You can find this at Symptom android project at activity org.coursera.symptom.activity.patient.CheckinActivity at line 286, method showCamera().

Another capability is at activity org.coursera.symptom.activity.doctor.ShowPhotoActivity that shows patient's photo to doctor and the user can make zoom and drag in the photo using Touch gestures. You can find this code in onTouchEvent method() at line 86 and in the inner classes ScaleGestureListener for control the in/out zoom of the photo and GestureListener to control the dragging at lines 221 and 251.

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

In fact all operations that access to server or database are done in background off the UI Thread. You can find a lot of examples along the code with different techniques. Basically there are three techniques that I have used in different scenarios:

1. **AysncTask**: I use AsyncTask at classes org.coursera.symptom.client.CallableTask and org.coursera.symtom.client.TaskCallback. This AsynTask is used by activities that want to access server or database. For example this scenario is used in activities: org.coursera.symptom.activity.LoginActivity at login() method at line 72 to makes a login process or at org.coursera.symptom.activity.doctor.PatientList at showPatientList() method at line 81 to download patient list in background or at org.coursera.symptom.activity.patient.CheckinActivity at method findMedications() at line to load medications from the local database. There are another places along the code at org.coursera.symptom.activity.doctor.CheckinDetailActivity, and org.coursera.symptom. activity.doctor.MedicationsFragment.
2. **Services**: When the doctor wants to see the patient's photo, it is used a service and a handler to communicate activity with the service. This allow app to download the photo in background. You can find this code at org.coursera.symptom.activity.doctor.ShowPhotoActivity at line 170 to start the service, at the MessangerHandler at line 148 that receive the response from the service and at the org.coursera.symptom.services.DownloadIntentService service that represents the service that download the photo from the server at downloadPhoto() method at line 249 and to send photo path to activity in order to draw the photo at sendPath()method at line 375.
3. **LoaderManager**: This is used to load data from database in a very efficient way. You can find an exemple at org.coursera.symptom.activity.doctor.CheckinListFragment class.

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.

You can find this at Symptom Android project in the org.coursera.symptom.activity.LoginActivity class at login() method at line 87 that loads who is this user from the serve. At SymptomServer spring project the getStatus() method at line 62 at org.coursera.symptomserver.controllers class loads patient's information from the server database using a coursera.symptomserver.services.PatientService class at getStatus() method at line 69 and getPatient() method at line 50. All patient's information is in the coursera.symptomserver.beans.jpa.Patient class that is loaded from database.

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

At Symptom Android project the patient can change the adjustable times at UserPreferences class that is located at org.coursera.symptom.activity.PreferenceActivity and in the inner class PatientPreferenceFragment. User can specify three parameters. The number of reminders (four or six), the first hour in a day to receive the first reminder and the last hour in a day to receive the last reminder. This allow users not to be bothered during the night.

The app defines the reminders as an alarm at org.coursera.symptom.activity.LoginActivity class when the user makes the login, at line 125 calling setCheckinReminderAlarm() method at org.coursera.symptom.utils.Utils class.

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 (items 4-8) at that date and time

At Symptom Android project the setCheckinReminderAlarm() method at line 265 at org.coursera.symptom.utils.Utils class, schedules a repeating alarm with the patient adjustable time. This method reads SharedPreferences object to know user preferences. When the alarm is fired it send the broadcast org.coursera.symptom.CHECKIN_REMINDER. This broadcast is received by a BroadcastReceiver org.coursera.symptom.receivers.NotificationCheckinReceiver that sends a notification to the patient mobile's notification area with an Intent associated to org.coursera.symptom.activity.patient.CheckinActivity class. When the user receive the notification and press this notification, the Check-in process is started. When the activity org.coursera.symptom.activity.patient.CheckinActivity starts builds a Wizard of all questions at buildWizard() method at line 148. This structure represents an ArrayList of org.coursera.symptom.utils.CheckinQuestion objects that contains questions and the answers for this questions that the user will respond during the check-in process. When the user finishes the Wizard the activity starts a org.coursera.symptom.services.SendIntentService service in order to sent this check-in to the server. In this service an org.coursera.symptom.orm.Checkin object is built at Checkin.createCheckin method()at line 74, and this check-in object is sent to the server associated to the patient at line 79 of this service.

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

This question is included at Symptom Android project at org.coursera.symptom.activity.patient.CheckinActivity class at line 151 in the buildWizard() method. Then the question is showed in the showQuestion() method at line 186. The method showQuestion is called from different places along this activity. First is called when the activity starts and the pain medications needs to be loaded from the local database at line 125 or in the onCreate() method at line 95 if the pain medications have been loaded before. Another places are in the onClick() method when the user presses some of the possible options for every question. This onClick() method controls what is the next question to be showed depending what option responds the user. And finally it is called in the onTimeSet() when the user selects a time in the TimePicker.

5. Check-In includes the question, “Did you take your pain medication?” to which a Patient can respond “yes” or “no”.

This question is included at Symptom Android project at org.coursera.symptom.activity.patient.CheckinActivity class at line 156 in the method buildWizard(). The process to show the question is detailed in the requirement 4 of this document.

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.”

This question is included at Symptom Android project at org.coursera.symptom.activity.patient.CheckinActivity class at lines 162 to 169 in the method buildWizard(). The process to show these questions is detailed in the requirement 4 of this document.

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.

At Symptom Android Project org.coursera.symptom.activity.patient.CheckinActivity class at line 217 at onClick() method checks if is needed to show a DatePickerFragment to allow user to select a Date. When the user selects a date in this dialog, the onDateSet() method at line 328 is called and this method shows a TimePickerFragment to allow user to select a time. When user select a time the onTimeSet() method is called. The date and time are saved in the CheckinQuestion object associated to the current question. To know if it is needed to show this DatePickerFragment, the CheckinQuestion object that represents every question of the check-in process contains a method isDateTimeRequired(). When the wizard is built at buildWizard() method at org.coursera.symptom.activity.patient.CheckinActivity the property dateTimeRequired is set to true at line 163.

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.

This question is included at Symptom Android Project at org.coursera.symptom.activity.patient.CheckinActivity class at line 170 at buildWizard() method. The process to show the question is detailed in the requirement 4 of this document.

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.

You can find this at Symptom Android Project at org.coursera.symptom.activity.LoginActivity class at login() method at line 87 that loads who is this user from local database if the user was authenticated at least one time in the past, or at line 90 calling the server to identified who is the user. At SymptomServer sprging project the method getStatus() at line 62 at org.coursera.symptomserver.controllers loads doctor's information from the server database using a coursera.symptomserver.services.DoctorService class at getStatus() method at line 80 and getDoctor() at line 60. All doctor's information is in the class coursera.symptomserver.beans.jpa.Doctor that is loaded from database.

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).

Doctors can monitor check-ins at Symptom Android Project at class org.coursera.symptom.activity.doctor.CheckinListActivity and org.coursera.symptom.activity.doctor.CheckinListFragment. These classes loads a patient's checkin list from the doctor mobile's local database using a LoaderManager<Cursor>. When the doctor selects one check-in row the activity org.coursera.symptom.activity.doctor.CheckinDetailActivity is started and it displays all check-in information. You can find this at onListItemClick() method at line 155.

These checkins are downloaded from the server and saved in the local database in the service org.coursera.symptom.services.DownloadIntentService class at downloadCheckins() method at line 221. This service is started by a repeating alarm that is scheduled when the App starts at class org.coursera.symptom.activity.LoginActivity at lines 127 and 141. The frequency of the repeating alarm is adjusted by user at User preferences screen. See class org.coursera.symptom.PrefefenceActivity

The doctors can monitor patient's pain evolution in the activity org.coursera.symptom.activity.doctor.MonitorActivity class that shows two graphics with the answers of How Bad is your mouth pain/sore throat? and Does your pain stop you from eating/drinking?

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).

At Symptom Android project at org.coursera.symptom.activity.doctor.PatientList class at line 147 a Dialog is showed when the user presses a menu option with magnifying glass icon. This dialog is an instance of an inner class SearchPatientDialogFragment defined at line 200. In this dialog the user can type patient's name and when they press the search button the method searchPatient() at line 159 is called. This method starts an AsyncTask to get patient's information from the server in the background and when the information arrives from the server it starts the activity org.coursera.symptom.activity.doctor.CheckinListActivity to see all patient's check-in data. This is done at line 176 calling the method startCheckinListActivity at class org.coursera.symptom.activity.doctor.BaseListActivity. This method save Patient object at org.coursera.symptom.SymptomApplication class and then starts CheckinListActivity.

When the activity CheckinListActivity is called it loads a CheckinListFragement that it loads check-in data from local database that is associated to patient recover from server. This is done at org.coursera.symptom.activity.doctor.CheckinListFragment fragment at line 175 at onCreateLoader() method.

At SymptomServer Spring project at org.coursera.symptomserver.controllers.SymptomController class the getPatient() method at line 136 is called when the doctor searches a patient in the app. This method gets patient from database calling getPatient() method from class org.coursera.symptomserver.services.DoctorService at line 121.

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).

At Symptom Android project at org.coursera.symptom.activity.doctor.MedicationsActivity and org.coursera.symptom.activity.doctor.activity.doctor.MedicationsFragment the doctor can manage patient's medications. This fragment uploads patient medication to the server calling at SymptomServer Spring project at org.coursera.symptomserver.controllers.SymptomController class, the createPatienMedication() method to create a new patient medication and the updateStatePatientMedication() method to activate or deactivate patient medication. These calls are located at lines 198, 215 and 224 lines at MedicationsFragment fragment class.

This patient medication is downloaded from patient's app using at Symptom Android project the org.coursera.symptom.services.DownloadIntentService service. This service is started via a repeating alarm that is scheduled when the app starts. The service download patient medications at method downloadMedications at line 197.

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.”

First of all, when the patient sends a Check-in to the server it calls at Symptom Server Spring project at org.coursera.symptomserver.controllers.SymptomController class the createCheckin() method at line 103. This method calls the createCheckin() method from org.coursera.symptomserver.services.DoctorService class at line 117. This method calculates if the patient needs medical attention to alert doctor because the patient experiences 12 hours of sever pain, 16 or more hours of moderate or severe pain and 12 hours of I can't eat. This is done calling isNeedsAlertDoctor method at line 124 that sets to true value the alertDoctor property of Checkin. This information is saved in the server database.

At Symptom Andproid project when the service org.coursera.symptom.services.DownloadIntentService is started by a fire alarm that is scheduled at org.coursera.symptom.activity.LoginActivity at lines 127 and 141, the method downloadCheckins() at line 221 is called. This method checks if any of the downloaded check-ins have the alertDoctor set to true value. This is done calling the method checkDoctorNotification() at line 292. If this happen then a notification is sent to the doctor mobile's notification area to alert doctor at line 320.

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

In Fact, all communication between app and server is done via https. The server is started at 8443 port to handle https protocol. This is done at SymptomServer Spring project at class org.coursera.symptomserver.Application at line 178.

At Symptom Project, the client connects via https building a Retrofit object of org.coursera.symptom.client.SymptomSvcApi class at line 66 from class org.coursera.symptom.client.SymptomSVC. This class uses at line 67 an http client object of org.coursera.symptom.client.EasyHttpClient class that uses a SSLSocketFactory for encrypt communication at inner class EasySSLSocketFactory at line 224.