Assignment 6

In this assignment, you will continue working on your code for Assignment 5 by adding server communication to your application.

The REST API to Use:

In this course, we use Backendless service as the RESTful server, which provides general REST APIs without needing any extra coding or settings in the server side.

If you've not done this before, first sign up as a new user in Backendless website. In your Backendless dashboard, you can create a new app. Then you can see your *Application ID* and *REST API key* which should be used in requests sent to Backendless.

The Backendless REST API addresses start with this format: *https://api.backendless.com/<app-id>/<api-key>/*. You can learn more about how to consume Backendless REST APIs by checking its quick start guide and the documentation if needed.

In Backendless dashboard, you can always check the data saved in your app by opening the *Data* tab in the left sidebar. If you want to test sending requests to Backendless outside Android code, you can use REST request builders like Postman or REST console of Backendless itself.

User Sign-Up:

Your application already has the ability to sign up new users. In this assignment, you should change the sign-up process so that the new user is saved to server instead of local database. Also user's email address should be used as username.

In order to save the new user to server, a POST request should be sent to */users/register* route of the Backendless server. The server assumes that you send the user data including at least *name*, *email* and *password* fields in the body of the request, with the same exact field names.

User Sign-In:

The first time user wants to login in the current device, the user is still not saved in local database. In this case, an online sign-in should be done with server by sending a POST request to */users/login* endpoint with *login* and *password* fields in the body, where login field should be the same as user's email address.

If the online sign-in is successful, the server returns the full user data, including the generated ID of the user in *objectId* field, and the authorization token in *user-token* field. Now the full user data can be saved to local database.

Because the ID fields of the main entities of your app are now generated in server, you should change your previous database schema to store the ID fields as non-automatic string values provided by server.

After this one-time online login, every time user wants to login in this device, because user is found in local database with a valid authorization token, an offline login can be performed without contacting the server.

Saving Changes to Tasks to Server:

Adding, updating and removing tasks should now be done in server. After saving to server successfully, the related task should be saved to or removed from local database.

For each of these operations, a request with appropriate method should be sent to */data/task* endpoint (e.g. POST for adding tasks or PUT for updating). Also all these requests should have the authorization token in their *user-token* header, so that server can confirm the user who is the owner of the task.

In case of adding or updating tasks, the server result includes ID of the task in *objectId* field, and ID of the user saving the data in *ownerId* field.

Notes about Implementation:

You should respect common implementation notes mentioned in your previous assignments in this assignment too. In addition:

1. Before working on this assignment, make sure the main functionalities of your Assignment 5 app work as expected, because in this assignment you should change and complete your code for the previous assignment.

2. Because in this assignment you may change the previous schema of your local database, you may need to uninstall your app for the previous assignment from your device, or at least clear its data, in order to prevent conflicts with previous database. The ideal way for handling changes in database schema is using Room migrations, but you're not required to use migration in this assignment.

3. For network requests, you should use the Volley networking library. For JSON processing, you should use the Gson library.

Notes about Submission:

1. You should submit a zipped file which includes both your root project folder and the final APK file of your project. You can find the generated APK file of your project in the path *YOURPROJECT/app/build/outputs/apk/* on disk.

2. In order to reduce the size of the project for submission, you can remove the two folders called *build* in your project folders, one is in the path *YOURPROJECT/build* and the other in the path *YOURPROJECT/app/build*. Make sure to get out the APK file before removing these folders.