CMPS 312 Mobile Application Development

LAB 8: Android Data management using SQLite and shared preferences

---

Objective

In this lab you will learn how data is persisted in android using,

1. Shared Preferences
2. Android SQLite Database

Overview

In the first part of the lab will have a short tutorial on how to,

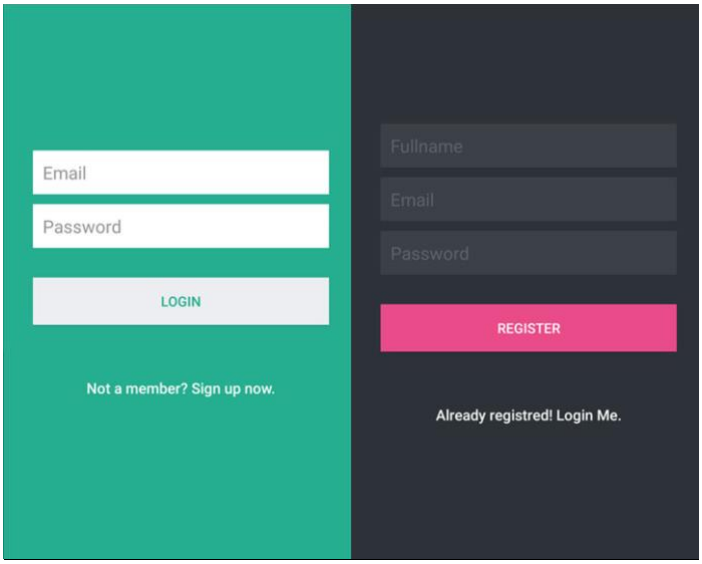
1. Read/Write data to shared preferences
2. How to do CRUD (create, read, update and delete) operations on SQLite database.

After the tutorial you will be required to implement,

1. A simple **login activity** which uses shared preferences to save its information
2. Create a complete employees application that persists its data using an SQLite Database.

PART A
Shared Preferences

In this part you are required to create a simple login/registration application that uses shared preferences to save user credentials.
Below is a sample Login Screen.



To accomplish this you should do the following.

1. Registration Screen
    A. Read all the user inputs such as, the First name , email and password from the edit text views
    B. When the user clicks on the Register Button
        I. Get the sharedPreferences object in the activity level.
        II. Get the editor of the shared preferences
        III. Save the user data in the shared preferences editor
        IV. Commit the editor
        V. Navigate the user back to the Login Screen
2. Login Screen
    A. Get the sharedPreferences object in the activity level
    B. Read back the user credentials that you saved in step 1.
    C. Compare the saved user credentials with what the user entered on the email and password edit text boxes.
    D. If they value match, then take the user to the home screen and display a welcome message. Otherwise, If the credentials don't match then Toast a message saying "wrong username/password, please try again".

PART B
Android SQLite Database

In this part you will create a simple employee management application. The application should allow adding, searching, updating, deleting and displaying records from employees' database. To accomplish this you should do the following steps.

1. Download the skeleton code from the blackboard. The skeleton code contains a basic UI design and the POJO employee class
2. Define the database **schema** and **contract** classes. The schema
    a. Create an interface and name it **DatabaseContract**
    b. Inside the **DatabaseContract** create an inner class and call it **EmployeeTable**
    c. **Let the EmployeeTable extend the BaseColumn Class**
    d. Add all the **column names** and **table name** inside the **EmployeeTable** class

```
static final String TABLE_NAME = "employees";
static final String COLUMN_NAME_EMP_ID = "emp_id";
static final String COLUMN_NAME_EMP_NAME = "name";
static final String COLUMN_NAME_PASSWORD = "password";
static final String COLUMN_NAME_EMAIL = "email";
static final String COLUMN_NAME_COUNTRY = "country";
```

Note: if you have more tables you can create a new class for each table.

3. Create the database helper class. This class will be responsible in creating all tables and updating the database.

a. Create a class and name it **DBHelper**. It should extend the **SqLiteOpenHelper** and it should implement the **DatabseContract** interface that you created in step 2.
b. Implement the onCreate , onUpdate and the Constructor.
c. In the onCreate create the employee table.
d. In the onUpgrade drop all tables and call the onCreate(db) method.

4. Create the Data access object class that will be responsible of handling all the CRUD operations in the database.
    a. Create a class called **EmployeeDAO** that implements the **DatabseContract** interface.
    b. Add the following two attributes

```
private DBHelper dbHelper;
private SQLiteDatabase db;
```

    c. Implement the following methods
        i. `public long addEmployee(Employee employee)`
        ii. `public long updateEmployee(Employee employee)`
        iii. `public long deleteEmployee(int emp_id)`
        iv. `public Employee getEmployee(int emp_id)`
        v. `public ArrayList<Employee> getAllEmployees()`
5. Link the **EmployeeDAO** class to the UI.