Android Studio

Based on IntelliJ IDEA, Android Studio is the official Integrated Development Environment (IDE) for Android app development. Android Studio adds even more features to IntelliJ's powerful code editor and developer tools, such as a flexible Gradle-based build system, a fast and feature-rich emulator, and a unified environment where you can develop for all Android devices, on top of IntelliJ's powerful code editor and developer tools.

To-Do List Application

In this project we fully coded a complete To-Do List application on Android Studio that allows the user to add, modify, & delete tasks from his to-do list, as well as registering an account & saving his list tasks to log in later & get his own list that he saved with a complete authentication process, where his private registered information will be saved & encrypted along with his list tasks.

The project consists of 5 different activities along with their XML layouts:

- Main Activity
- Add Activity
- Update Activity
- Register Activity
- Log In Activity

Main Activity

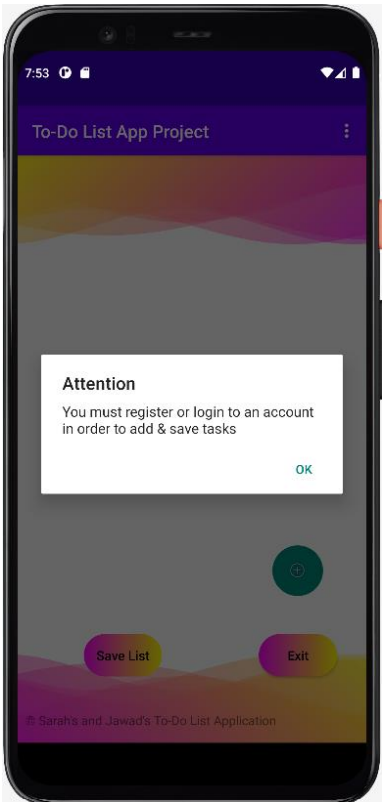
Starting with the main activity, we had all our attributes initialized at the top

- The ArrayList we are going to use.
- The ListView.
- A menu button for the login page.
- Shared Preferences file to save list information.
- A MultiWaveHeader for decorations.
- And the add & save buttons.

```
ArrayAdapter<String> adapter;
static ArrayList<String> list = new ArrayList<>();
ListView lv;
protected Menu menu;
private static SharedPreferences sharedPreferences;
private static final String sharedPreFileName = "UserListItems.txt";

MultiWaveHeader waveFooter, waveHeader;

com.google.android.material.floatingactionbutton.FloatingActionButton Addbtn;
Button Savebtn;
```

Next up, we have our onCreate method, where we created an adapter for the ListView & connected them together, along with adding an OnItemClickListener which contains an intent to take us to the update activity, where when an item from the list gets clicked, we go to the update activity and be able to modify or delete it.
The ability to click the add & save buttons will beset to false at the start, and a dialog will pop up at the start of the application telling the user that in order to use those buttons he will have to login to his account or register of it doesn't have one.
Some coloring codes related to the MultiWaveHeader were also added.

```
//onCreate----------------------------------------------------------------------------------------

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    lv = findViewById(R.id.lv);
    adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, list);

    lv.setAdapter(adapter);
    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

            Intent intent=new Intent(getApplicationContext(),UpdateActivity.class);
            String theTask=list.get(i).toString();
            intent.putExtra( name: "theTask",theTask);
            startActivityForResult(intent, requestCode: 2);
        }
    });


    //WaveMotion------------
    waveFooter = findViewById(R.id.wave_footer);
    waveFooter.setVelocity(1);
    waveFooter.setProgress(1);
    waveFooter.isRunning();
    waveFooter.setGradientAngle(45);
    waveFooter.setWaveHeight(40);
    waveFooter.setStartColor(Color.MAGENTA);
    waveFooter.setCloseColor(Color.YELLOW);

    Addbtn = findViewById(R.id.Floatingbtn);
    Addbtn.setEnabled(false);

    Savebtn = findViewById(R.id.Savebtn);
    Savebtn.setEnabled(false);

    Dialog Dialog = new Dialog();
    Dialog.show(getSupportFragmentManager(), tag: "Dialog");
```



Right after, we made couple methods for the menu button to take us to the login page.
An encryption & a decryption methods & created a save button which is going to encrypte & save our data from the list to a shared preference.
And a AddTask button which it basically moves us to the AddActivity class.

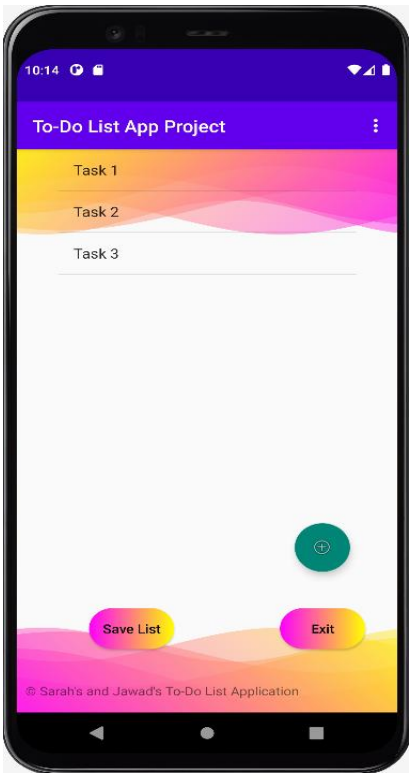
```
//MenuBtn-------------------------------------------------------------------------------------
@Override
public boolean onMenuItemClick(MenuItem item) {
    Intent intent = new Intent( packageContext: this, LoginActivity.class);
    startActivityForResult(intent,  requestCode: 3);
    return true;
}

public void refreshMenu() {
    MenuItem Option = menu.findItem(R.id.first_action);
    Option.setTitle("Log In");
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    this.menu = menu;
    refreshMenu();
    return true;
}
```

```
//encryption & decryption-----------------------------------------------------------------------------------------
public static String encrypt(String input) {
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}

//SaveBtn---------------------------------------------------------------------------------------------------
public void onSave(View view)  {

    sharedPreferences = getSharedPreferences(sharedPreFileName, MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();

    for (int i = 0; i < list.size(); i++) {
        String data = list.get(i);
        editor.putString(encrypt( input: "List Item"), encrypt(data));
        editor.commit();
    }
    Toast.makeText( context: this,  text: "List Items Have Been Saved!", Toast.LENGTH_SHORT).show();
}

//AddTaskBtn------------------------------------------------------------------------------------------------
public void AddTask(View view) {

    //Toast.makeText(this,"Please Login First ",Toast.LENGTH_SHORT).show();
    Intent intent = new Intent( packageContext: this, AddActivity.class);
    startActivityForResult(intent,  requestCode: 1);

}
```

OnActivityResult is our next method that is responsible for the actions that going to be taken after returning from a specific activity according to a requestCode & a resultCode that they get assigned to, on requestCode 1 we will be adding a tank, on requestCode 2 we will be updating a task, and on requestCode 3 we are going to load the user list (if he had any) and allow him to use the adding & save buttons after he successfully logs in.

```
//OnActivityResult-------------------------------------------------------------------------------
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == 1) {
        if(resultCode == 1){String newTask = data.getStringExtra( name: "newTask");
        list.add(newTask);
        adapter.notifyDataSetChanged();}

        if(resultCode == 2){
            adapter.notifyDataSetChanged();
        }
    } //End of Request Code 1

    if (requestCode == 2) {

        String UpdateTask = data.getStringExtra( name: "UpdateTask");
        String theTask = data.getStringExtra( name: "theTask");
        if(resultCode == 1){
            if (theTask != UpdateTask){
                list.remove(theTask);
                list.add(UpdateTask);
                adapter.notifyDataSetChanged();
            }else adapter.notifyDataSetChanged(); } //End of Result Code 1

    if(requestCode == 3){
        if(resultCode == 1){
            sharedPreferences = getSharedPreferences(sharedPreFileName,MODE_PRIVATE);
            int S = sharedPreferences.getAll().size();
            list.clear();
            for (int i = 0; i < S; i++) {
                String loadedDataEn = sharedPreferences.getString(encrypt( input: "List Item"), encrypt( input: "No Data Loaded"));
                String loadedData = decrypt(loadedDataEn);
                list.add(loadedData);
                adapter.notifyDataSetChanged();
            }
            Addbtn = findViewById(R.id.Floatingbtn);
            Addbtn.setEnabled(true);

            Savebtn = findViewById(R.id.Savebtn);
            Savebtn.setEnabled(true);
            }//End of Result Code 1
        }//End of Request Code 3
}//End onActivityResult


//ExitBtn-----------------------------------------------------------------------------------------
public void onExit(View view) { finish(); }

}
```

` By that we have our MainActivity class finished with the following results:



AddActivity

Moving on to our AddAcitivity, it was mainly about two buttons:

An Adding button, that adds the written tasks to the list.

And a cancel button, in case the user changed his mind in adding a task.

```
public void onAdd(View view)
{
    EditText et=findViewById(R.id.editText);
    Intent intent=getIntent();
    String newTask=et.getText().toString();
    intent.putExtra( name: "newTask",newTask);
    setResult( resultCode: 1, intent);
    finish();
}

public void onCancel(View view) {
    Intent intent=getIntent();
    setResult( resultCode: 2, intent);
    finish();
}
```

Update Activity

Next in our UpdateAcitivty, right in the onCreate method we took the values from the intent from MainActivity having the key “theTask” to get what was written in the box that the user clicked to modify.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_update);

    ed = findViewById(R.id.editText);
    Intent intent = getIntent();
    String theTask = intent.getStringExtra( name: "theTask");
    ed.setText(theTask);
```

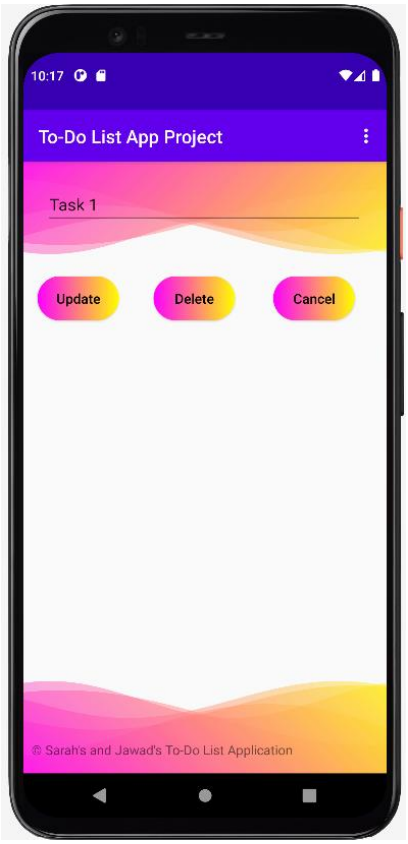
It contains 3 buttons:

- Button onUpdate to modify the task and send it back to MainActivity after modification.
- OnCancel button, same as the on in AddActivity.
- And onDelete button that deletes the task and send back an empty slot to main Activity.

```
public void onUpdate(View view) {
    EditText ed=findViewById(R.id.editText);
    Intent intent=getIntent();
    String newTask=ed.getText().toString();
    intent.putExtra( name: "UpdateTask",newTask);
    setResult( resultCode: 1, intent);
    finish();
}
```

```
public void onCancel(View view) {
    Intent intent=getIntent();
    setResult( resultCode: 3, intent);
    finish();
}
```

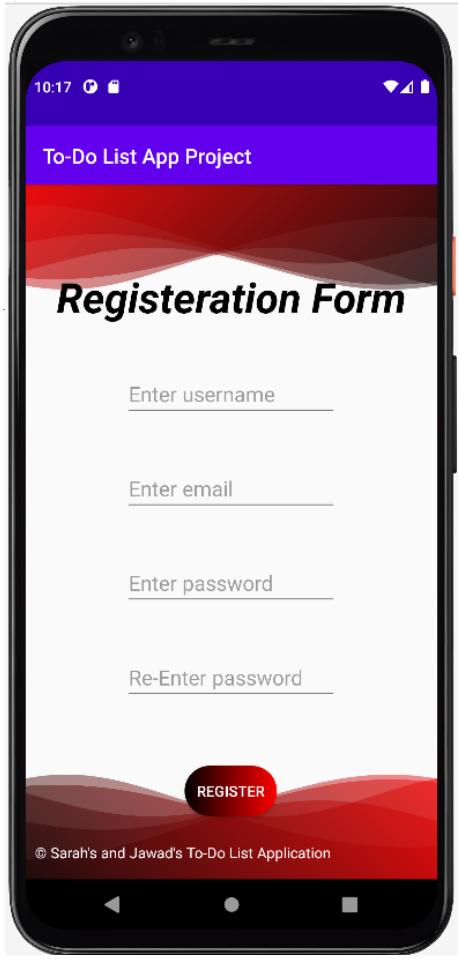
```
public void onDelete(View view) {
    EditText ed=findViewById(R.id.editText);
    Intent intent=getIntent();
    String newTask=ed.getText().toString();
    intent.putExtra( name: "DeletedTask",newTask);
    setResult( resultCode: 2, intent);
    finish();

}
```



Register Activity

This activity job is to take 4 different inputs from the user, a username, an email, a password, and a password configuration, encrypts & saves them all in a shared preference file. The username, email, and password validity are all checked through validatePassword & validateEmail methods.

```
public boolean validatePassword(String password, String passwordcon) {
    Pattern uppercase = Pattern.compile("[A-Z]");
    Pattern lowercase = Pattern.compile("[a-z]");
    Pattern digitcase = Pattern.compile("[0-9]");


    if (password.isEmpty()) {
        Toast.makeText( context: this,  text: "Password can not be empty!", Toast.LENGTH_SHORT).show();
        return false;
    } else if (!lowercase.matcher(password).find()) {
        Toast.makeText( context: this,  text: "Password Must Contain Lowercase!", Toast.LENGTH_SHORT).show();
        return false;
    } else if (!uppercase.matcher(password).find()) {
        Toast.makeText( context: this,  text: "Password Must Contain Uppercase!", Toast.LENGTH_SHORT).show();
        return false;
    } else if (!digitcase.matcher(password).find()) {
        Toast.makeText( context: this,  text: "Password Must Contain Digits!", Toast.LENGTH_SHORT).show();
        return false;
    } else if (password.length() <= 8) {
        Toast.makeText( context: this,  text: "Password Length Must be 8 characters or more!", Toast.LENGTH_SHORT).show();
        return false;
    } else if (!password.equals(passwordcon)) {
        Toast.makeText( context: this,  text: "Passwords must match!", Toast.LENGTH_SHORT).show();
        return false;
    } else
    return true;
}

public boolean validateEmail(String email) {
    if(!email.isEmpty() && Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        return true;
    }else{
        Toast.makeText( context: this,  text: "Invalid Email Address!", Toast.LENGTH_SHORT).show();
        return false;
    }
}
//encryption----------------------------------------------------------------------------------------
public static String encrypt(String input) {
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}


public void onRegister(View view) {
    sharedPreferences = getSharedPreferences(RegDetails, MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();

    String data = username.getText().toString();
    String data2 = email.getText().toString();
    String data3 = password.getText().toString();
    String data4 = passwordconf.getText().toString();

    if (validatePassword(data3, data4) == true && validateEmail(data2) == true) {
        Toast.makeText( context: this,  text: "Register Success!", Toast.LENGTH_SHORT).show();
        editor.putString(encrypt( input: "UserName"), encrypt(data));
        editor.putString(encrypt( input: "Email"), encrypt(data2));
        editor.putString(encrypt( input: "Password"), encrypt(data3));
        editor.putString(encrypt( input: "Password Config"), encrypt(data4));
        editor.commit();
        finish();
    }
```



Login Activity

Lastly, we have our LoginAcitivty, that are going to be taken two inputs from the user, his user name and password, and matched with the values from register activity. We have 2 buttons, one for login and one for sign up, the one for sign up contains an intent that takes us to register activity, while the login button takes two encrypted values from the shared preference file that we created in RegActivity (The UserName, and the Password), decrypts them, & then it makes sure that the two values entered matches the data saved from registering.

After the authentication process, if they match & everything is good, the activity will finish and load the saved list for the user (If there were any saved lists from before) & enables the save & add tasks buttons.

```
//onLogin-----------------------------------------------------------------------------------------------------------
public void onLogin(View view){
    et = findViewById(R.id.editTextTextPersonName);
    et2 = findViewById(R.id.editTextTextPersonName2);

    sharedPreferences=getSharedPreferences(RegDetails,MODE_PRIVATE);

    String loginusernameEn = sharedPreferences.getString(encrypt( input: "UserName"),encrypt( input: "null"));
    String loginpasswordEn = sharedPreferences.getString(encrypt( input: "Password"),encrypt( input: "null"));
    String loginpassword = decrypt(loginpasswordEn);
    String loginusername = decrypt(loginusernameEn);

      if(et.getText().length()==0){
        et.setError("Please Enter Your Username");

    }
      else if(et2.getText().length()==0){
        et2.setError("Please Enter Your Password");

    }
      else if(!et2.getText().toString().equals(loginpassword)){
        Toast.makeText( context: this, text: "Password is incorrect ",Toast.LENGTH_SHORT).show();

    } else if (!et.getText().toString().equals(loginusername)){
        Toast.makeText( context: this, text: "Username is incorrect",Toast.LENGTH_SHORT).show();
        }

      else{
          Intent intent=getIntent();
          setResult( resultCode: 1, intent);
          finish();
      }
//encryption & decryption-------------------------------------------------------------------------
public static String encrypt(String input) {
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```