

# **Login & Weather Application, JSON Parser**

## **Homework 3**

1. [Overview](#)
2. [Specifications](#)
3. [Perquisites](#)
4. [Documentation](#)
  - a. [Creating an account](#)
  - b. [Sign in](#)
  - c. Home Screen
  - d. [Manage Profile](#)
  - e. Favourite Links
  - f. Weather
  - g. Tennis Players
5. [Tests](#)
  - a. [Test 1- Sign Up](#)
  - b. [Test 2- Sign In](#)
  - c. [Test 3 - Profile](#)
  - d. Test 4- Favourite Links
  - e. Test 5- Weather
  - f. Test 6 - Tennis Players
  - g. Known Bugs
6. [Functions](#)
  - a. [Sign In](#)
  - b. [Sign Up](#)
  - c. [Profile](#)
  - d. Weather Activity - Structure
  - e. Tennis Players - JSON
7. [Appendix 1](#)
8. Appendix2

### **1. Overview**

The purpose of this assignment was to build a simple application for mobile devices which contains a *log in* interface and displays *weather* informations. The two major platforms on the market, nowadays, are Android and iOS.

Android, is the world's most popular mobile platform, powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content. That's why we chose **Android SDK** to develop our application.

The application was built using the **Eclipse IDE**, with the **ADT plug-in** for **Android SDK**.

## 2. Specifications

- a. The application will display a Log In interface
- b. The user will be able to log in, or create an account (the informations of the account are stored in a databae)
- c. As an authenticated user, one can:
  - check the profile
  - check favourite links
  - check the weather
- d. An authenticated user can modify the profile/change the password
- e. An authenticated user can add favourite links
- f. An authenticated user can change the city displayed in weather info

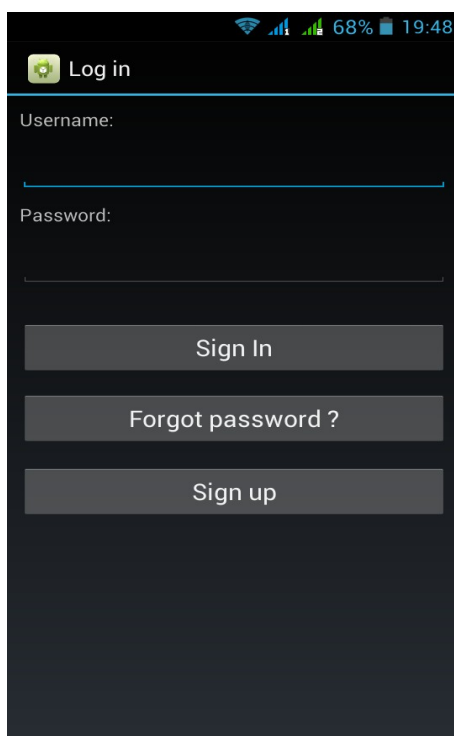
## 3. Perquisites

- a. A device with at least Android 2.3.1 (Froyo)
- b. The device must have an internet connection
- c. The application is target for the Android 4.0.4, API Level 15
- d. 1,5 MB of free space

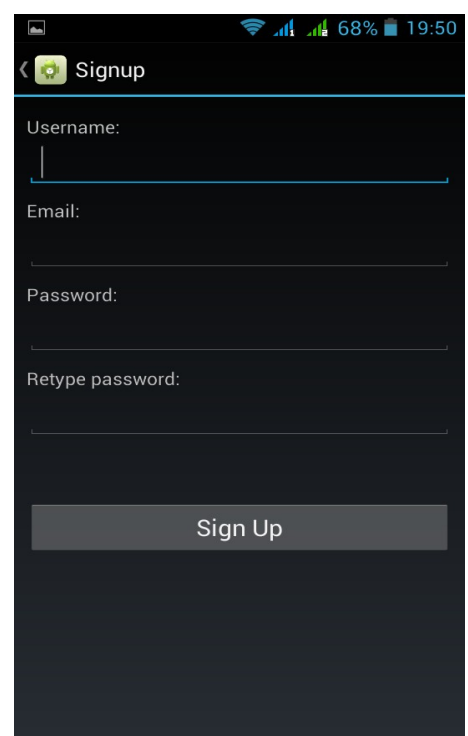
## 4. Documentation

## a. Creating an account

Launch the application on your phone and choose **Sign Up** from the menu (Fig1). A new activity will appear where you can fill your information (Fig2). Press **Sign Up** again to create the account. If everything was right, a message will be prompted and the application will return to the **Log In** activity (Fig1).



(Fig 1)



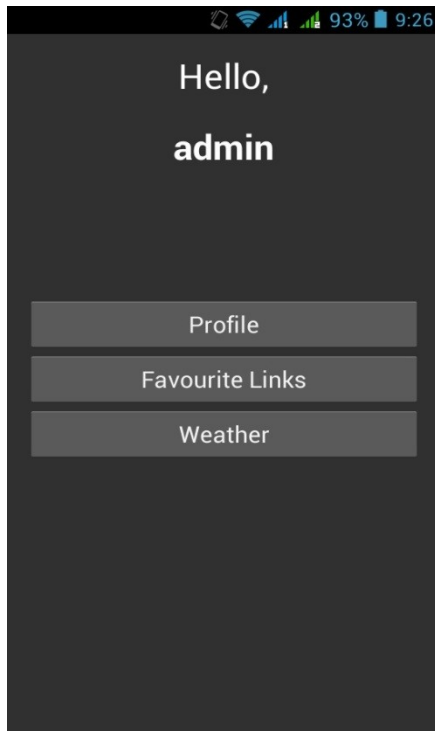
(Fig 2)

## b. Sign In

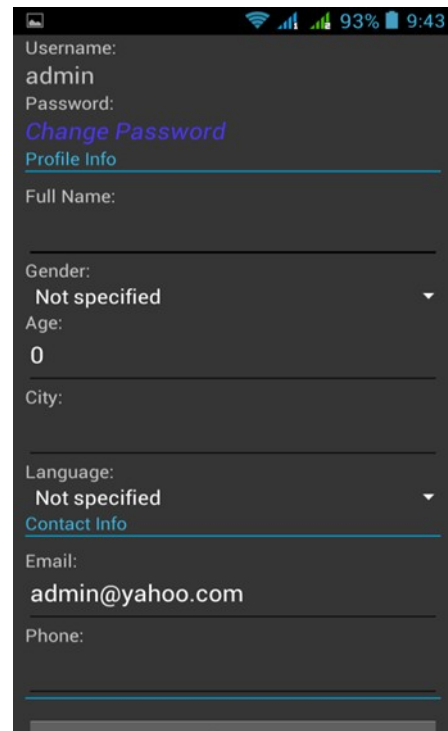
If you already have an account, fill in the username and the password in the **Log In** activity (Fig 1).

## c. Home Screen

After successfully signing in into the application, the home screen is displayed (Fig 3), where you can select one of these actions: check/modify your Profile, check/add your Favourite Links or check the Weather.



(Fig 3)

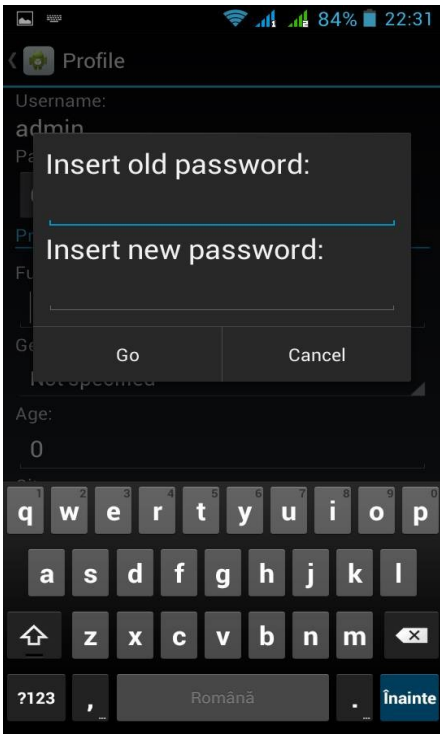


(Fig 4)

## d. Manage Profile

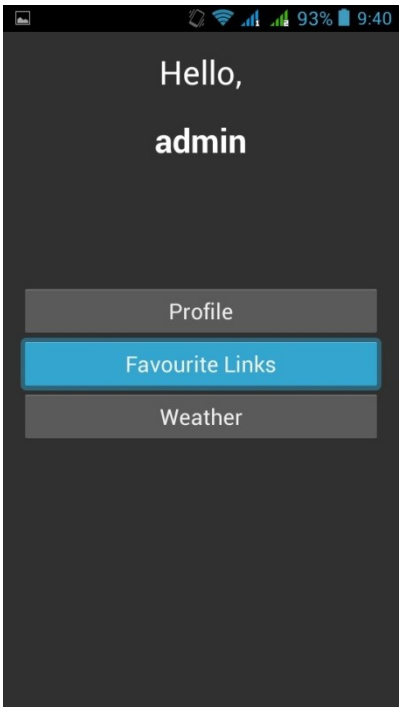
If you have a new account, entering the Profile screen (Fig 4), most of the fields will be empty. You can complete your profile by filling into the text views for each field, or by choosing from the list boxes. Then, hit the **Save** button at the end in order to save your data. **Caution:** The updates will not be saved after you close the application without pressing **Save** button.

In order to change, your password tap **Change Password**. A dialog will appear, as in (Fig 5). Introduce your old password and your new password and hit OK. It is not required to hit **Save** after this operation.

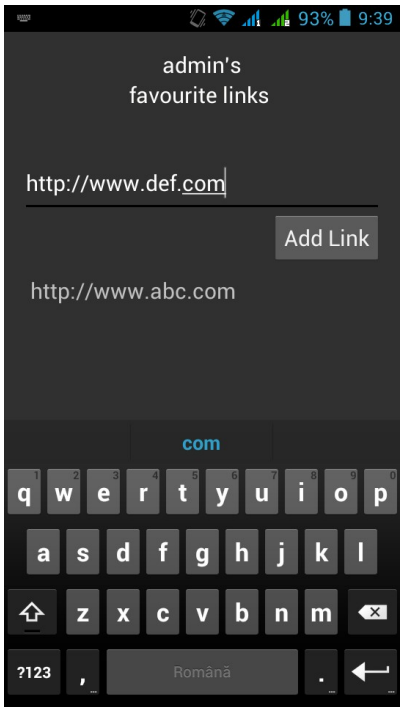


(Fig 5)

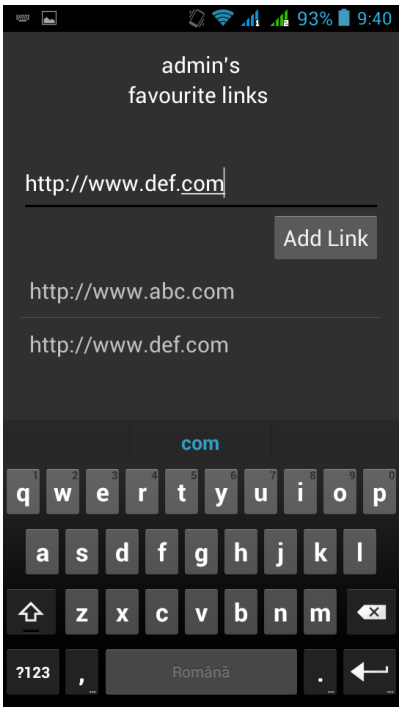
e. Favourite Links



(Fig 6)



(Fig 7)



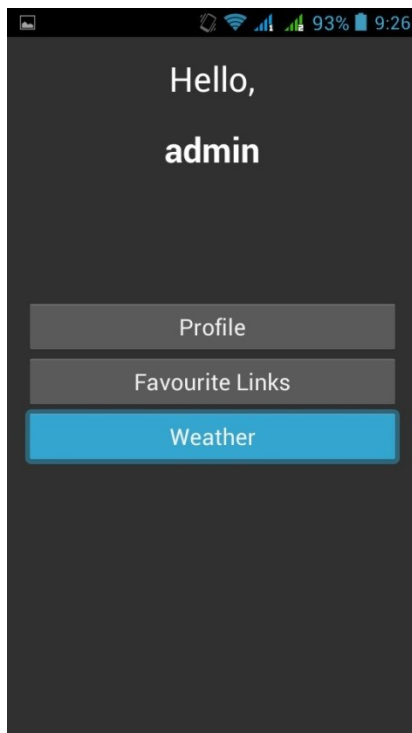
(Fig8)

Selecting Favourite Links button from the Home screen (Fig 6), we enter in Favourite Links screen, where we can see the links we added (Fig 7).

In order to add a new link, we must enter the link in the format “http://www.example.com” and press the Add Link button. The new link is displayed in our list (Fig 8).

## f. Weather

Selecting Weather button from the home screen (Fig 9), a screen with weather information is displayed (Fig 10 - city name, sky condition, humidity, pressure and temperature).

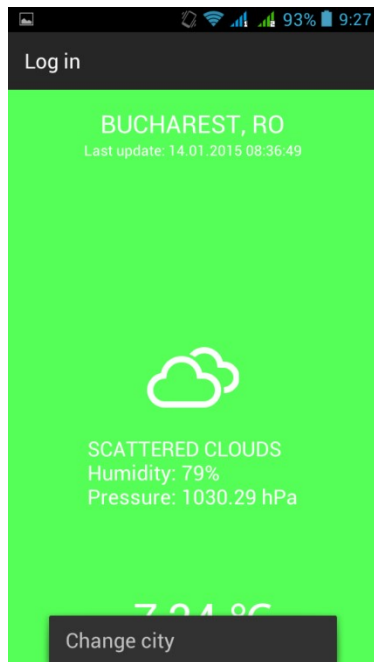


(Fig 9)

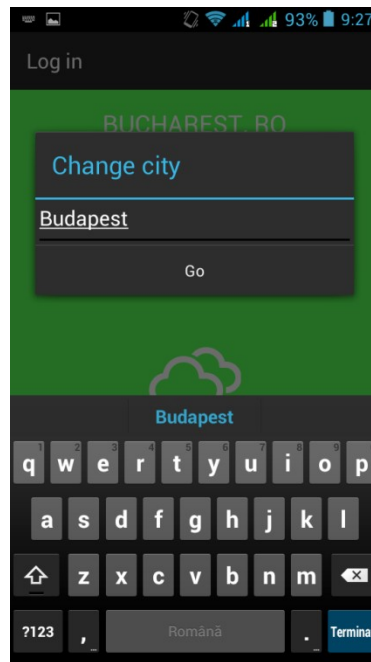


(Fig 10)

Pressing the mobile's device Settings button, appears the “Change city” option (Fig 11). In the dialog box shown we can enter the new city name (Fig 12) and the new information is displayed (Fig 13).



(Fig 11)



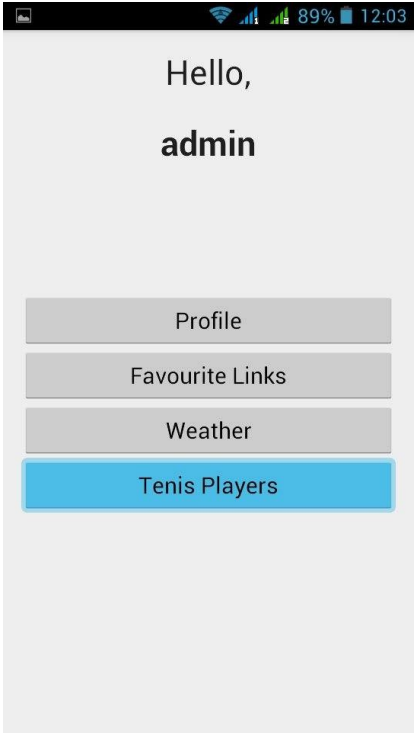
(Fig 12)



(Fig 13)

## g. Tennis Players

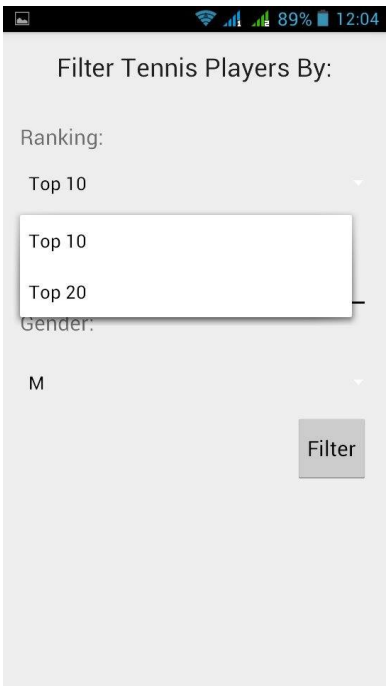
For the HW3 we slightly modified the design (the colors). In the Home Screen we added the Tennis Players button (Fig 14) which displays a filter form depending on some characteristics (Top, Age, Gender - Fig 15a-b-c).



(Fig 14)



(Fig 15a)



(Fig 15b)



(Fig 15c)

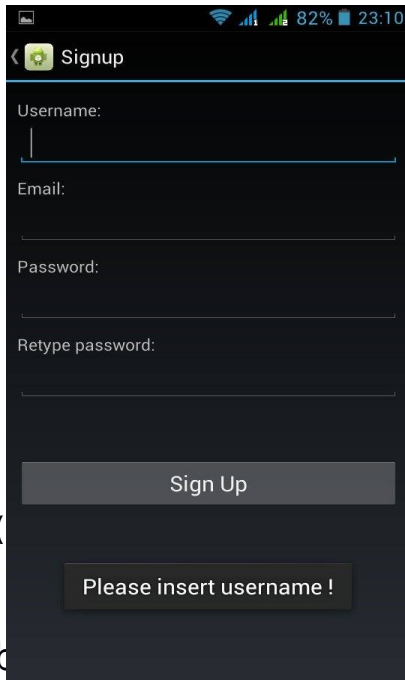
## 5. Tests

### a. Test 1 - Sign up

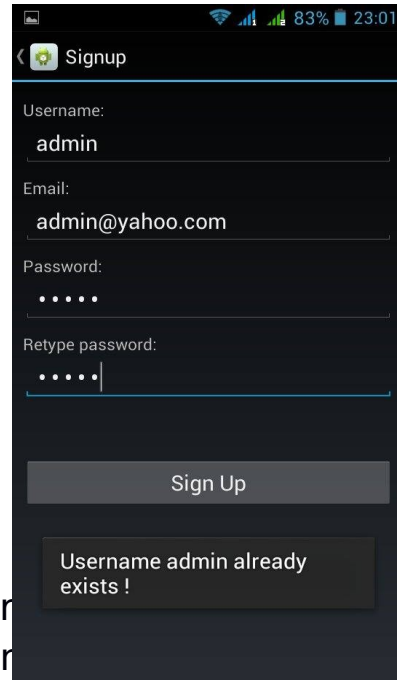


A number of constraints have been applied to ensure the data validation.

If the user tries to register with an empty field user name, password or email, a message will be prompted and the account won't be created (Fig 16).



The screenshot shows the 'Signup' screen with the following fields: Username (empty), Email (empty), Password (empty), and Retype password (empty). A message box at the bottom states 'Please insert username !'. The status bar at the top shows 82% battery and 23:10.



The screenshot shows the 'Signup' screen with the following fields: Username (filled with 'admin'), Email (filled with 'admin@yahoo.com'), Password (filled with dots), and Retype password (filled with dots). A message box at the bottom states 'Username admin already exists !'. The status bar at the top shows 83% battery and 23:01.

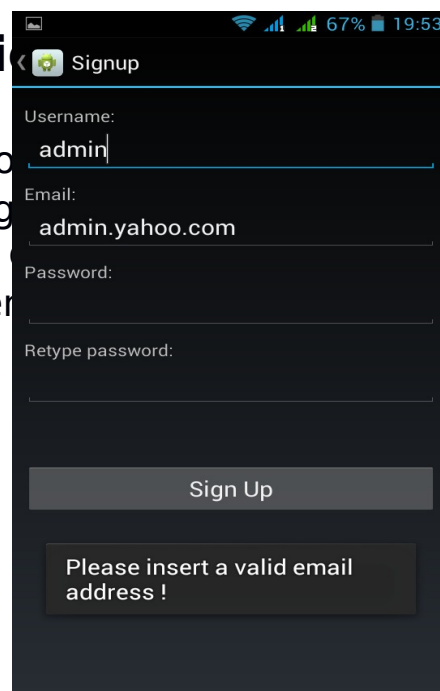
Typing error, it has to be created and a message will be prompted.

To ensure, the singularity of the username, the database is checked before. If there is another account with the same username, the account won't be created (Fig 17).

## b. Test 2 - Si

If you try to register with an empty field, password or username, a message

If the username (password) does not match any recording in the database, then (Fig 18) will be displayed.



The screenshot shows the 'Signup' screen with the following fields: Username (filled with 'admin'), Email (filled with 'admin.yahoo.com'), Password (empty), and Retype password (empty). A message box at the bottom states 'Please insert a valid email address !'. The status bar at the top shows 67% battery and 19:53.

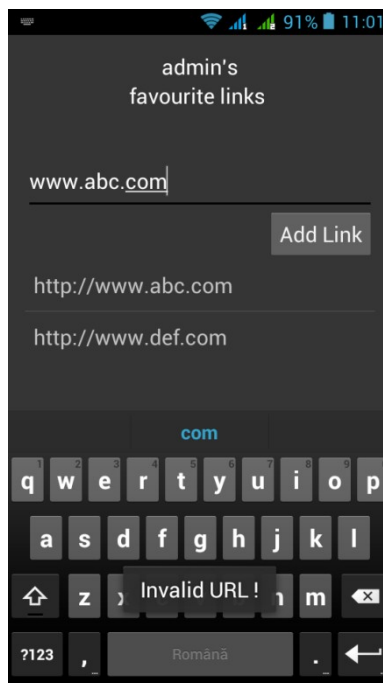
(Fig 18)

### **c. Test 3- Profile**

The user can insert the into the Profile view, and save by hitting the save button. No email validation has been implemented yet. The password can be changed from the Dialog in (Fig 5). The old password is checked to correspond to the one in the database.

### **d. Test 4- Favourite Links**

The user can insert favourite links only in the specified format. If the link doesn't start with "http://... " a toast with the message "Invalid URL" will be displayed (Fig 19).



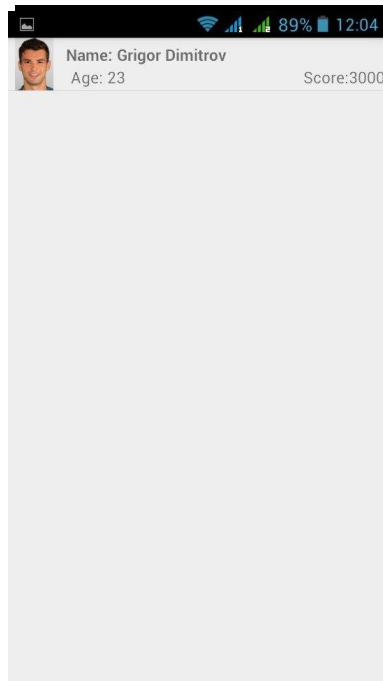
(Fig 19)

### **e. Test 5- Weather**

The new city can be typed only in letters (Fig 12) or we can enter its ZIP code.

### **f. Test 6- Tennis Players**

After selecting the characteristics from the form, we press the button Filter. A list with the tennis players that fit our search (Top 10, Age < 25, Male) is shown (Fig 20).



(Fig 20)

## **g. Known Bugs**

- i. No data will be saved if suddenly the internet connection fails, no data saved locally
- ii. No email validation in the Profile activity(not implemented yet)
- iii. Forgot Password button is not implemented yet
- iv. Can't delete/modify favourite links

## **6. Functions**

### **a.Sign in Activity(LoginActivity.java) - Functions**

#### **void onCreate() method :**

- is called when the application is launched. The user interface is built here based on the corresponding XML file

- the connection with the parse.com database is created here
- controls are connected to the widgets in the interface

### **void signin() method:**

- called by clicking on the **Sign In** button
- checks the data inserted and looks for the user in the database
- if the data is valid launches the Profile Activity sending the username via an Intent

### **void signup() method:**

- called by clicking on the **Sign Up** button
- opens the Sign Up Activity

See full code with comments in Appendix 1.

## **b. Sign Up Activity(SignupActivity.java) - Functions**

### **void onCreate() method :**

- is called when the application is launched. The user interface is built here based on the corresponding XML file
- the connection with the parse.com database is created here
- controls are connected to the widgets in the interface

### **void signup() method:**

- called by clicking on the **Sign Up** button
- validates the data inserted into the fields
- performs regular expression validation on the email
- checks for the existence of the introduced username
- creates a new account and saves it in the database

See full code with comments in Appendix 1.

## **c. Profile Activity(ProfileActivity.java) - Functions**

### **void onCreate() method :**

- is called when the application is launched. The user interface is built here based on the corresponding XML file
- the connection with the parse.com database is created here
- gets the username sent by the **Sign In** activity via intent
- queries the database for the given username'
- controls are connected to the widgets in the interface
- updates the fields with the information from the database

### **void popDialog() method:**

- called by clicking on the **Change Password** button
- opens an AlertDialog with two fields for old an new password
- queries the table for the old password to compare it to the inserted one
- updates the new password(no empty field allowed)

### **void saveData() method:**

- called by clicking on the **Save** button
- updates the database for each field with the values inserted by the user
- return to Sign In acitivity

See full code with comments in Appendix 1.

## **d.Weather Activity- Structure**

- **activity\_weather.xml** - It should already have a FrameLayout. We add an extra property to change the color of the background.
- **fragment\_weather.xml** - we add five TextView tags to show the following information:
  - o city and country
  - o current temperature

- an icon showing the current weather condition
  - a timestamp telling the user when the weather information was last updated
  - more detailed information about the current weather, such as description and humidity
- **strings.xml** - This file contains the strings used in our app as well as the Unicode character codes that we'll use to render the weather icons. The application will be able to display eight different types of weather conditions.
- **menu/weather.xml** - The user should be able to choose the city whose weather they want to see. We edit *menu/weather.xml* to add an item for this option.
- **RemoteFetch.java** - This class is responsible for fetching the weather data from the OpenWeatherMap API.

We use the *URLConnection* class to make the remote request. The OpenWeatherMap API expects the API key in an HTTP header named *x-api-key*. This is specified in our request using the *setRequestProperty* method.

We use a *BufferedReader* to read the API's response into a *StringBuffer*. When we have the complete response, we convert it to a *JSONObject* object.

The JSON data contains a field named *cod*. Its value is *200* if the request was successful. We use this value to check whether the JSON response has the current weather information or not.
- **CityPreference.java** - The user shouldn't have to specify the name of the city every time they want to use the app. The app should remember the last city the user was interested in. We do this by making use of *SharedPreferences*. However, instead of directly accessing these preferences from our *Activity* class, it is better to create a separate class for this purpose.

We create a new Java class and name it *CityPreference.java*. To store and retrieve the name of the city, create two methods *setCity* and *getCity*. The *SharedPreferences* object is initialized in the constructor.
- **WeatherFragment.java** - This fragment uses *fragment\_weather.xml* as its layout. Declare the five *TextView* objects and initialize them in the *onCreateView* method. Declare a new *Typeface* object named *weatherFont*. The *Typeface* object will point to the web font you downloaded and stored in the *assets/fonts* folder.

We will be making use of a separate *Thread* to asynchronously fetch data from the OpenWeatherMap API. We cannot update the user

interface from such a background thread. We therefore need a *Handler* object, which we initialize in the constructor of the *WeatherFragment* class.

Initialize the *weatherFont* object by calling *createFromAsset* on the *Typeface* class. We also invoke the *updateWeatherData* method in *onCreate*.

In *updateWeatherData*, we start a new thread and call *getJSON* on the *RemoteFetch* class. If the value returned by *getJSON* is *null*, we display an error message to the user. If it isn't, we invoke the *renderWeather* method.

Only the main Thread is allowed to update the user interface of an Android app. Calling *Toast* or *renderWeather* directly from the background thread would lead to a runtime error. That is why we call these methods using the handler's *post* method.

The *renderWeather* method uses the JSON data to update the *TextView* objects. The weather node of the JSON response is an array of data. In this tutorial, we will only be using the first element of the array of weather data. At the end of the *renderWeather* method, we invoke *setWeatherIcon* with the id of the current weather as well as the times of sunrise and sunset. Setting the weather icon is a bit tricky, because the *OpenWeatherMap* API supports more weather conditions than we can support with the web font we're using.

We use the sunrise and sunset times to display the sun or the moon, depending on the current time of the day and only if the weather is clear.

Finally, we add a *changeCity* method to the fragment to let the user update the current city. The *changeCity* method will only be called from the main *Activity* class.

- **WeatherActivity.java** - During the project's setup, Eclipse populated *WeatherActivity.java* with some boilerplate code. We replace the default implementation of the *onCreate* method.

Next, edit the *onOptionsItemSelected* method and handle the only menu option we have. All you have to do here is invoke the *showInputDialog* method.

In the *showInputDialog* method, we use *AlertDialog.Builder* to create a *Dialog* object that prompts the user to enter the name of a city. This information is passed on to the *changeCity* method, which stores the name of the city using the *CityPreference* class and calls the *Fragment*'s *changeCity* method.

## e. Tennis Players - JSON



JSON stands for JavaScript Object Notation. It is an independent data exchange format and is the best alternative for XML.

In the Tennis Players activity, we parse the JSON object and fetch the data. The players list is built based on the data in the JSON object.

## 7. Appendix 1

### LoginActivity.java

```
package com.example.md_hw1;

import java.util.List;

import com.parse.FindCallback;
import com.parse.Parse;
import com.parse.ParseAnalytics;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.parse.PushService;
import com.parse.ParseException;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class LoginActivity extends Activity {
    private EditText usernameEdit;
    private EditText passwordEdit;
    private final String USER_TABLE = "Profiles";
    public final static String EXTRA_MESSAGE =
"com.example.myfirstapp.MESSAGE";
    private String message;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        //The arguments are the APP_ID, and the CLient_ID from parse.com
        Parse.initialize(this, "udumMDhauwhzIXHbWK1Bm3twUIP63F4y2Eq1BGVO",
"FXKo0ICrczFFrYLMmvn6qRg3bB4rLCkzKdkS1qOU");
        ParseAnalytics.trackAppOpened(getIntent());
        //Get username and password editText from layout
```

Miu Catalin Marian  
Neagu Petru Liviu

```
        usernameEdit=(EditText)findViewById(R.id.usernameEdit);
        passwordEdit=(EditText)findViewById(R.id.passwordEdit);
    }
    public void signin(View view){
        //Fetch the username and password from the edittext
        String username=usernameEdit.getText().toString();
        String password=passwordEdit.getText().toString();
        if(username.length()==0)
            Toast.makeText(getApplicationContext(), "Please insert
username !", Toast.LENGTH_SHORT).show();
        else
            if(password.length()==0)
                Toast.makeText(getApplicationContext(), "Please insert
password !", Toast.LENGTH_SHORT).show();
            else
            {
                //Initiate a parse query
                ParseQuery<ParseObject> query = ParseQuery.getQuery(USER_TABLE);

                //Select where username is @username and password is @password
                query.whereEqualTo("username", username);
                query.whereEqualTo("password", password);

                //make the query
                try{
                    List<ParseObject> userList=query.find();
                    //if the username was found open Profile activity
                    if(userList.size()>0)
                    {

                        message=username;
                        Toast.makeText(getApplicationContext(),"Redirecting...",
Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(this, Profile.class);
                        //send the user name via intent to be used in profile
                        activity

                        intent.putExtra(EXTRA_MESSAGE, message);
                        startActivity(intent);
                    }
                    else
                        Toast.makeText(getApplicationContext(), "Invalid
username/password !", Toast.LENGTH_SHORT).show();

                }
                catch(ParseException e){
                    Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
                }
            }
        passwordEdit.setText("");
    }
    public void signup(View view){
        //start the signup activity
        Intent intent = new Intent(this, Signup.class);
```

Miu Catalin Marian  
Neagu Petru Liviu

```
        this.startActivity(intent);  
    }  
}
```

## Signup.java

```
package com.example.md_hw1;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import com.parse.FindCallback;  
import com.parse.Parse;  
import com.parse.ParseAnalytics;  
import com.parse.ParseException;  
import com.parse.ParseObject;  
import com.parse.ParseQuery;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Signup extends Activity {  
    private EditText usernameEdit;  
    private EditText passwordEdit;  
    private EditText repasswordEdit;  
    private EditText emailEdit;  
  
    private final String USER_TABLE = "Profiles";  
    private String username, password, email;  
    //regular expression pattern for email validation  
    private static final String EMAIL_PATTERN =  
        "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"  
        + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";  
  
    private ParseObject users, contacts, profiles; //the object handles the  
    table users  
  
    private Pattern pattern;  
    private Matcher matcher;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_signup);  
        //Initialize PARSE  
        //The arguments are the APP_ID, and the Client_ID from parse.com
```

Miu Catalin Marian  
Neagu Petru Liviu

```
Parse.initialize(this, "udumMDhauwhzIXHbWK1Bm3twUIP63F4y2Eq1BGV0",
"FXKoOICrczFFrYLMmvn6qRg3bB4rLCkzKdkS1qOU");
ParseAnalytics.trackAppOpened(getIntent());

//Create a new ParseObject that will handle the users table
users = new ParseObject(USER_TABLE);
//Get username and password editText from layout
usernameEdit=(EditText)findViewById(R.id.username);
passwordEdit=(EditText)findViewById(R.id.password);
repasswordEdit=(EditText)findViewById(R.id.repassword);
emailEdit=(EditText)findViewById(R.id.email);

//Regular expression email validator
pattern = Pattern.compile(EMAIL_PATTERN);
}

public void signup(View view){
    //Fetch the username and password from the edittext
    username=usernameEdit.getText().toString();
    password=passwordEdit.getText().toString();
    email=emailEdit.getText().toString();
    if(username.length()==0)
        Toast.makeText(getApplicationContext(), "Please insert
username !", Toast.LENGTH_LONG).show();
    //if it DOES NOT validates the reg expression for email
    if(pattern.matcher(email).matches()==false)
        Toast.makeText(getApplicationContext(), "Please insert a valid
email address !", Toast.LENGTH_LONG).show();
    else
        if(password.length()==0)
            Toast.makeText(getApplicationContext(), "Please insert
password !", Toast.LENGTH_LONG).show();
        else
            if(!password.equals(repasswordEdit.getText().toString()))
                Toast.makeText(getApplicationContext(), "Passwords are not
the same", Toast.LENGTH_LONG).show();
            else
            {
                //Make a query to check wether the user already exists
                ParseQuery<ParseObject> query = ParseQuery.getQuery(USER_TABLE);
                //Select where username is @username
                query.whereEqualTo("username",username);
                query.findInBackground(new FindCallback<ParseObject>() {
                    public void done(List<ParseObject> userList, ParseException
e) {
                        if (e == null) {
                            if(userList.size()>0)
                                Toast.makeText(getApplicationContext(), "Username
"+username+" already exists !", Toast.LENGTH_LONG).show();
                            else
                            {
                                Toast.makeText(getApplicationContext(), "Account
created !", Toast.LENGTH_SHORT).show();
                                users.put("username", username);
                                users.put("password", password);
                                users.put("username",username);
                            }
                        }
                    }
                });
            }
        }
    }
}
```

Miu Catalin Marian  
Neagu Petru Liviu

```
        users.put("Email",email);
        users.saveInBackground();
        finish();
    }
} else {
    Toast.makeText(getApplicationContext(),
e.getMessage(), Toast.LENGTH_SHORT).show();
}
});
}
}
}
```

## Profile.java

```
package com.example.md_hw1;

import java.util.List;

import com.parse.Parse;
import com.parse.ParseAnalytics;
import com.parse.ParseException;
import com.parse.ParseObject;
import com.parse.ParseQuery;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class Profile extends Activity {
    private TextView
usernameView, fullnameView, ageView, cityView, emailView, phoneView, passwordView;
    private Spinner genderView, languageView;
    private final String USERS_TABLE ="Profiles";
    private String ID,password;
    private ParseObject user;
    @Override
```

Miu Catalin Marian  
Neagu Petru Liviu

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profile);
    //The arguments are the APP_ID, and the Client_ID from parse.com
    Parse.initialize(this, "udumMDhauwhzIXHbWK1Bm3twUIP63F4y2Eq1BGV0",
"FXKo0ICrczFFrYLMmvn6qRg3bB4rLCKzKdkS1q0U");
    //Get textviews and spinners
    genderView = (Spinner) findViewById(R.id.gender);
    // Create an ArrayAdapter using the string array and a default
    spinner layout
    ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
        R.array.genders, android.R.layout.simple_spinner_item);
    // Specify the layout to use when the list of choices appears

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
);
    // Apply the adapter to the spinner
    genderView.setAdapter(adapter);

    //Init language spinner
    languageView=(Spinner)findViewById(R.id.language);
    ArrayAdapter<CharSequence>
ladapter=ArrayAdapter.createFromResource(this,
        R.array.languages, android.R.layout.simple_spinner_item);
    // Specify the layout to use when the list of choices appears

ladapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_ite
m);
    // Apply the adapter to the spinner
    languageView.setAdapter(ladapter);

    usernameView=(TextView)findViewById(R.id.username);
    fullnameView=(TextView)findViewById(R.id.fullname);
    passwordView=(TextView)findViewById(R.id.password);
    ageView=(TextView)findViewById(R.id.age);
    cityView=(TextView)findViewById(R.id.city);
    emailView=(TextView)findViewById(R.id.email);
    phoneView=(TextView)findViewById(R.id.phone);
    // Get the message from the intent
    Intent intent = getIntent();
    String username = intent.getStringExtra(LoginActivity.EXTRA_MESSAGE);
    //Query the database for the rest of the information
    ParseQuery<ParseObject> query = ParseQuery.getQuery(USERS_TABLE);
    //Select where username is @username and password is @password
    query.whereEqualTo("username", username);
    try{
        List<ParseObject> userList=query.find();
        //Fetch the data from the database into the fields on the
activity
        ParseObject user=userList.get(0);
        ID=user.getObjectId();
        usernameView.setText(user.getString("username"));
        password=user.getString("password");
        fullnameView.setText(user.getString("fullname"));
        genderView.setSelection(user.getInt("Gender"));
```

Miu Catalin Marian  
Neagu Petru Liviu

```
        ageView.setText(Integer.toString(user.getInt("Age")));
        cityView.setText(user.getString("City"));
        languageView.setSelection(user.getInt("Language"));
        emailView.setText(user.getString("Email"));
        phoneView.setText(user.getString("Phone"));
    }
    catch(ParseException e){
        Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
    }

}

public void popDialog(View view){
    // get prompts.xml view
    user = ParseObject.createWithoutData(USERS_TABLE, ID);
    LayoutInflater li = LayoutInflater.from(this);
    View promptsView = li.inflate(R.layout.password, null);
    final AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
    alertDialogBuilder.setView(promptsView);

    final EditText oldpass = (EditText) promptsView
        .findViewById(R.id.oldpass);
    final EditText newpass = (EditText) promptsView
        .findViewById(R.id.newpass);

    // set dialog message
    //add button and action listeners
    alertDialogBuilder
        .setCancelable(false)
        .setNegativeButton("Go",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,int id) {
                    //Chheck the old passwrod to correspond to the inserted
one
                    //Make sure new password is not null
                    if(oldpass.getText().toString().equals(password) &&
newpass.getText().length()>0){
                        user.put("password",newpass.getText().toString());
                        user.saveInBackground();}
                    else
                        Toast.makeText(getApplicationContext(), "Invalid
old/new password", Toast.LENGTH_SHORT).show();

                })
            })
        .setPositiveButton("Cancel",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,int id) {
                    dialog.dismiss();
                }
            })
        );
};
```

Miu Catalin Marian  
Neagu Petru Liviu

```
        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();
    }
    public void saveData(View view){
        user = ParseObject.createWithoutData(USERS_TABLE, ID);

        // Update database
        user.put("fullname", fullnameView.getText().toString());
        user.put("Gender", genderView.getSelectedItemPosition());
        user.put("Age", Integer.parseInt(ageView.getText().toString()));
        user.put("City", cityView.getText().toString());
        user.put("Language", languageView.getSelectedItemPosition());
        user.put("Email", emailView.getText().toString());
        user.put("Phone", phoneView.getText().toString());
        // Save
        Toast.makeText(getApplicationContext(), "Saving...",
Toast.LENGTH_SHORT).show();
        try {
            user.save();
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        finish();
    }
}
```

**9. Appendix 2:** See the full code on GitHub.