**University of Politehnica Bucharest**

**Faculty of Engineering in Foreign Languages**



**MOBILE DEVICES PROJECT**

**~Log in Application ~**

Professor: **Dragos Stoica**

Student**: Stefana Alina Simionescu**

**Victor Balau**

Group**: 1232E**

**CONTENTS**

1. INTRODUCTION
   1. Analisys +Specifications
   2. Strategy
2. DESCRIPTION AND USER’S GUIDE

2.1 What is a Android Application?

2.2 User’s Guide

1. DIAGRAM OF OUR PROJECT
2. TESTING (FUNCTIONALITY OF CLASSES)
3. CONCLUSIONS
4. REFERENCES

1. INTRODUCTION

**1.1 Analisys + Specifications**

Logging is a fundamental part of applications. Every application has a varying flavor of logging mechanism. A well designed logging system is a huge utility for system administrators and developers, especially the support team. Logs save many valuable hours for both the support team or developers. As users execute programs at the front end, the system invisibly builds a vault of event information (log entries) for system administrators and the support team.

Android applications operate within a shared resource environment, and the performance of your application can be impacted by how efficiently it interacts with those resources in the larger system. Applications also operate in a multithreaded environment, competing with other threaded processes for resources, which can cause performance problems that are hard to diagnose.

Analyze the program and design the classes and the relation between the classes, this step is most important because if you don’t prevision and don’t look-ahead or don’t pay attention to all facilities that the application can be have and make your project, probably during to develop the application you may encounter with the inspected problem.

**1.2 Strategy**

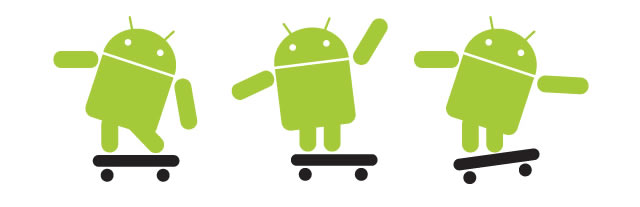
The objective behind this project is to simplify the idea of a contact database. It represents an user friendly service that offers mobile solutions for people who need to store contacts on the go on a secured web server, being able to access the information stored on any other pc/laptop/mac which has an internet connection.

Designing a good logging and instrumentation strategy is important for the security and reliability of your application. Failure to do so can make your application vulnerable to repudiation threats, where users deny their actions, and log files may be required for legal proceedings to prove wrongdoing. You should audit and log activity across the layers of your application, which can help to detect suspicious activity and provide early indication of a serious attack. Auditing is usually considered most authoritative if the audits are generated at the precise time of resource access, and by the same routines that access the resource. Instrumentation can be implemented by using performance counters and events to give administrators information about the state, performance, and health of an application. Consider the following guidelines when designing a logging and instrumentation strategy:

Design a centralized logging and instrumentation mechanism that captures system- and business-critical events. Avoid logging and instrumentation that is too fine grained, but consider additional logging and instrumentation that is configurable at run time for obtaining extra information and to aid debugging.

2. DESCRIPTION AND USER’S GUIDE

**2.1 What is a Android Application?**



Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touchscreen mobile devices such as smartphones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear). The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touchscreen input, it also has been used in game consoles, digital cameras, and other electronics.

Android is developed in private by Google until the latest changes and updates are ready to be released, at which point the source code is made available publicly.[91] This source code will only run without modification on select devices, usually the Nexus series of devices. The source code is, in turn, adapted by OEMs to run on their hardware.[92] Android's source code does not contain the often proprietary device drivers that are needed for certain hardware components.

Applications ("apps"), that extend the functionality of devices, are developed primarily in the Java programming language[65] using the Android software development kit (SDK). The SDK includes a comprehensive set of development tools,[66] including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) plugin.

**2.2 User’s Guide**

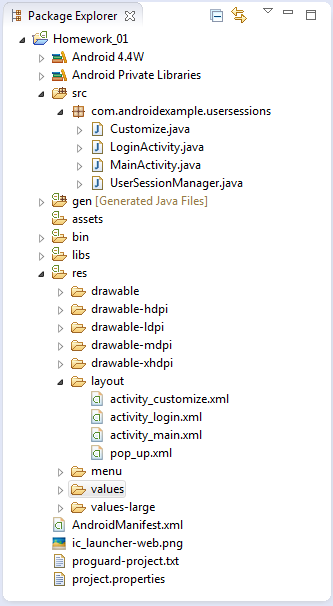
* Create a project namely Homework\_01 on the Eclipse SDK Manager and add all the following files to the project.

*Customize.java*

*LoginActivity.java*

*MainActivity.java*

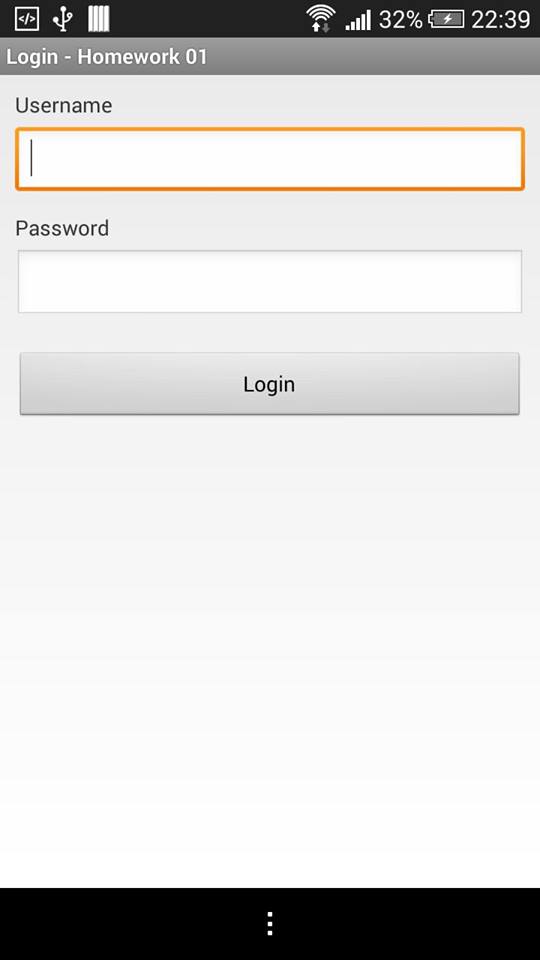
*UserSessionManager.java*

****

* LOGIN

It is a first form, which contains 2 fields: Username and Password. Logging in is usually used to enter a specific page, which trespassers cannot see. Once the user is logged in, the login token may be used to track what actions the user has taken while connected to the site.

Logging in or on and signing in or on is the process by which individual access to a computer system is controlled by identifying and authenticating the user through the credentials presented by the user.



Entering the Username: Alina with the password: 0000, or Victor respectively 123456, we have the opportunity to access the application, passing on the next page.

A username is almost always paired with a [password](http://www.techterms.com/definition/password). This username/password combination is referred to as a [login](http://www.techterms.com/definition/login), and is often required for users to log in to websites. For example, to access your e-mail via the Web, you are required to enter your username and password. Once you have logged in, your username may appear on the screen, but your password is kept secret. By keeping their password private, people can create secure accounts for various websites. Most usernames can contain letters and numbers, but no spaces.

* MAIN MENU

The **main menu** is the starting point; for example, in Windows, the Start**menu** is the **main menu**.Each application also has a **main menu** that appears as a row of options in a title bar.

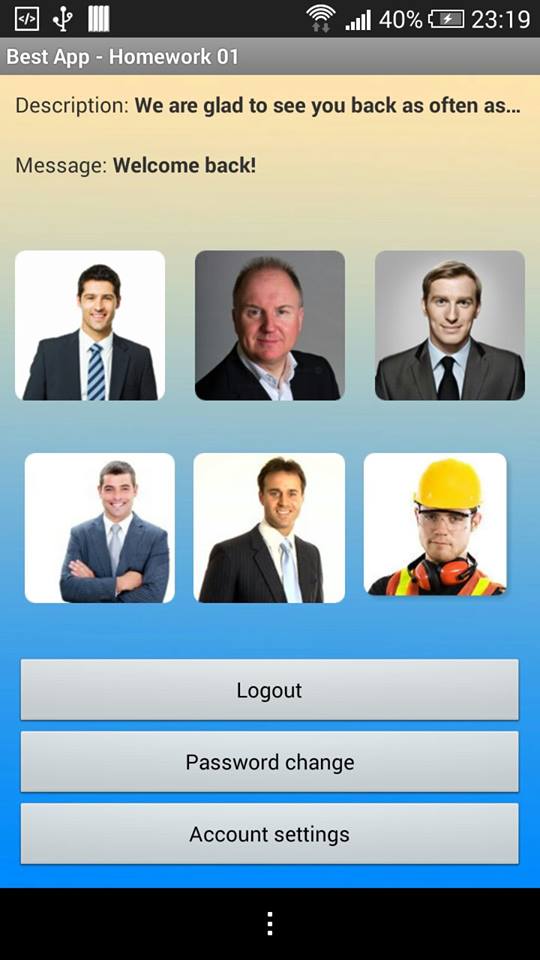
In our case, the user have the possibility to access different options:

1. Log Out

2. Password Change

3.Account Settings

We also have a welcome message and a description to create a pleasant environment for users.

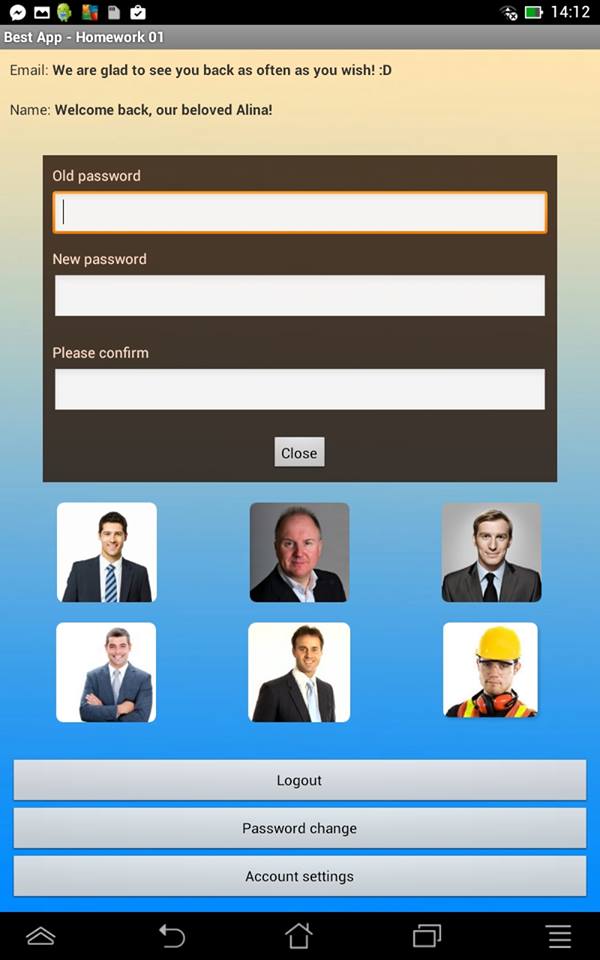


* LOGOUT

 When is using log out button, the operation works totally fine. Meaning that it does actually log the user off. Using this button, the user is directed to the original page (log in).

* PASSWORD CHANGE

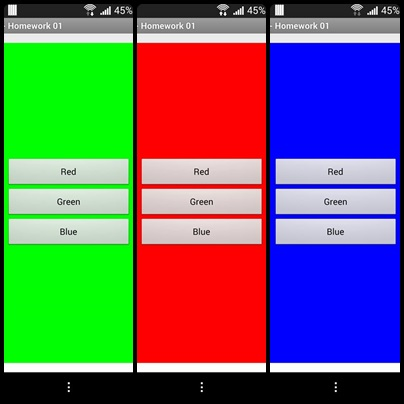
 A password is a word or string of characters used for user authentication to prove identity or access approval to gain access to a resource (example: an access code is a type of password), which should be kept secret from those not allowed access. The use of passwords is known to be ancient. Sentries would challenge those wishing to enter an area or approaching it to supply a password or *watchword*, and would only allow a person or group to pass if they knew the password.



* ACCOUNT SETTINGS

Android's purpose is to establish an open platform for developers to build innovative apps. The Android Compatibility program defines the technical details of the Android platform and provides tools used by OEMs to ensure that developers' apps run on a variety of devices. The Android SDK provides built-in tools that developers use to clearly state the device features their apps require. And Google Play shows apps only to those devices that can properly run them.

A mobile phone is a highly personal, always-on, always-present gateway to the Internet. We haven't met a user yet who didn't want to customize it by extending its functionality. That's why Android was designed as a robust platform for running aftermarket applications.



3. DIAGRAM OF OUR PROJECT



Values:

-color

-dimens

-strings

-styles

Menu:

-activity\_main

Layout:

-activity\_customize

-activity\_login

-activity\_main

-pop\_up

AndroidManifest

4. TESTING (FUNCTIONALITY OF CLASSES)

* Customize is the class which contains the method for creating buttons (Red, Green, Blue), and the method OnClick- in this way the background will change the color.

|  |  |
| --- | --- |
| Functions and constructor | Type and access level |
| onCreate | public void |
| onClick | public void |

Explications for Java codes

**import android.app.Activity-** Once we have imported a class from a different package, you will be able to refer to that class without specifying the fully qualified path for a specific class. So, after we have added the statement ‘import android.app.Activity;’ at the top of the file, you will be able to derive your class from Activity like ‘public class Customize extends Activity’ without having to say ‘public class Customize extends android.app.Activity’.

***Method OnCreate*** – Is called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

This method create buttons (Red, Blue, Green).

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_customize*);

red = (Button) findViewById(R.id.*red*);

green = (Button) findViewById(R.id.*green*);

blue = (Button) findViewById(R.id.*blue*);

mLayout = (LinearLayout) findViewById(R.id.*lyt\_customize*);

red.setOnClickListener(**new** OnClickListener() {

***Method OnClick*** – Every Button is styled using the system's default button background, which is often different from one device to another and from one version of the platform to another. If you're not satisfied with the default button style and want to customize it to match the design of your application, then you can replace the button's background image with a state list drawable. A state list drawable is a drawable resource defined in XML that changes its image based on the current state of the button. Once you've defined a state list drawable in XML, you can apply it to your Button with the android:background attribute.

This method refers to called the parameters when a view has been clicked.

**public** **void** onClick(View arg0) {

mLayout.setBackgroundColor(Color.*RED*);

}

});

green.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

mLayout.setBackgroundColor(Color.*GREEN*);

}

});

blue.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

mLayout.setBackgroundColor(Color.*BLUE*);

}

});

* LoginActivity - the act of logging in to a database, mobile device, or computer,especially a multiuser computer or a remote or networked computersystem. a username and password that allows a person to log in to a computer system, network, mobile device, or user account.

|  |  |
| --- | --- |
| Functions and constructor | Type and access level |
| onCreate | public void |
| onClick | public void |

Explications for Java codes

**import android.app.Activity-** Once we have imported a class from a different package, you will be able to refer to that class without specifying the fully qualified path for a specific class. So, after we have added the statement ‘import android.app.Activity;’ at the top of the file, you will be able to derive your class from Activity like ‘public class LoginActivity extends Activity’ without having to say ‘public class LoginActivity extends android.app.Activity’.

***Method OnCreate*** – Is called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

This method creates 2 fields for Username and Password.

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_login*);

// User Session Manager

session = **new** UserSessionManager(getApplicationContext());

// get Email, Password input text

txtUsername = (EditText) findViewById(R.id.*txtUsername*);

txtPassword = (EditText) findViewById(R.id.*txtPassword*);

Toast.*makeText*(getApplicationContext(),

"User Login Status: " + session.isUserLoggedIn(),

Toast.*LENGTH\_LONG*).show();

// User Login button

btnLogin = (Button) findViewById(R.id.*btnLogin*);

// Login button click event

btnLogin.setOnClickListener(**new** View.OnClickListener() {

***Method OnClick*** – Every Button is styled using the system's default button background, which is often different from one device to another and from one version of the platform to another. If you're not satisfied with the default button style and want to customize it to match the design of your application, then you can replace the button's background image with a state list drawable. A state list drawable is a drawable resource defined in XML that changes its image based on the current state of the button. Once you've defined a state list drawable in XML, you can apply it to your Button with the android:background attribute.

Using this method, the user have the possibility to log in as a user in his account.

**public** **void** onClick(View arg0) {

// Get username, password from EditText

String username = txtUsername.getText().toString();

String password = txtPassword.getText().toString();

// Validate if username, password is filled

**if**(username.trim().length() > 0 && password.trim().length() > 0){

// For testing puspose username, password is checked with static data

// username = admin

// password = admin

**if**(username.equals("admin") && password.equals("admin")){

// Creating user login session

// Statically storing name="Android Example"

// and email="androidexample84@gmail.com"

session.createUserLoginSession("Android Example", "androidexample84@gmail.com");

// Starting MainActivity

Intent i = **new** Intent(getApplicationContext(), MainActivity.**class**);

i.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);

// Add new Flag to start new Activity

i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

startActivity(i);

finish();

}**else** **if**(username.equals("victor") && password.equals("123456")){

// Creating user login session

// Statically storing name="Android Example"

// and email="androidexample84@gmail.com"

session.createUserLoginSession("Android Example", "androidexample84@gmail.com");

// Starting MainActivity

Intent i = **new** Intent(getApplicationContext(), MainActivity.**class**);

i.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);

// Add new Flag to start new Activity

i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

startActivity(i);

finish();

}

**if**(username.equals("alina") && password.equals("0000")){

// Creating user login session

// Statically storing name="Android Example"

// and email="androidexample84@gmail.com"

session.createUserLoginSession("Welcome back!", "We are glad to see you back as often as you wish! :D Established in 1988, Quanta Computer is the largest notebook computer ODM company in the world. With leading technology and strong R&D capability, Quanta has become a leader in hi-tech markets and the best partner providing quality design and manufacturing services to top-notched brands worldwide for technology products.");

// Starting MainActivity

Intent i = **new** Intent(getApplicationContext(), MainActivity.**class**);

i.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);

// Add new Flag to start new Activity

i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

startActivity(i);

finish();

}**else** {

// username / password doesn't match

Toast.*makeText*(getApplicationContext(), "Username/Password is incorrect", Toast.*LENGTH\_LONG*).show();

}

}**else**{

// user didn't entered username or password

Toast.*makeText*(getApplicationContext(), "Please enter username and password", Toast.*LENGTH\_LONG*).show();

}

}

});

* MainActivity – is the class when the user can access his account. In fact, the architecture of the system is quite different - the system loads Activities whenever the user starts an applicaiton/widged.

We need to identify our main Activity class - it should be in our project folder under src/some/package/name/OurActivity.java (or similar). We have learn how Activities work and how user interfaces are created and loaded both via XML and programmatically, though.

|  |  |
| --- | --- |
| Functions and constructor | Type and access level |
| onCreate | public void |
| onClick | public void |
| PopupWindow (constructor) | private |
| initiatePopupWindow | public void |

Explications for Java codes

**import java.util.HashMap -** The **java::import** command provides a means to specify Java class names in a shortened format. This functionality is the equivalent of the *import* statement in Java. Once imported, the Java class**java.util.Hashtable** could be specified with the shortened name **Hashtable**. Each *class* argument is checked to ensure that it is the name of a accessible class and does not conflict with another class imported from another package. The optional **-package** *pkg* arguments are used to indicate the package that each *class* argument is imported from. If the **-package** *pkg* arguments are given, none of the*class* arguments can include a package specifier, meaning the '.' character must not appear in a *class* name.

***Method OnCreate*** – Is called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

// Session class instance

session = **new** UserSessionManager(getApplicationContext());

TextView lblName = (TextView) findViewById(R.id.*lblName*);

TextView lblEmail = (TextView) findViewById(R.id.*lblEmail*);

// Button logout

btnLogout = (Button) findViewById(R.id.*btnLogout*);

btnCustomize = (Button) findViewById(R.id.*btnCustomize*);

btnPass = (Button) findViewById(R.id.*btnPass*);

Toast.*makeText*(getApplicationContext(),

"User Login Status: " + session.isUserLoggedIn(),

Toast.*LENGTH\_LONG*).show();

// Check user login

// If User is not logged in , This will redirect user to LoginActivity.

**if** (session.checkLogin())

finish();

// get user data from session

HashMap<String, String> user = session.getUserDetails();

// get name

String message = user.get(UserSessionManager.*KEY\_NAME*);

// get email

String description = user.get(UserSessionManager.*KEY\_EMAIL*);

// Show user data on activity

lblName.setText(Html.*fromHtml*("Message: <b>" + message + "</b>"));

lblEmail.setText(Html.*fromHtml*("Description: <b>" + description + "</b>"));

btnLogout.setOnClickListener(**new** View.OnClickListener() {

***Method OnClick*** – Every Button is styled using the system's default button background, which is often different from one device to another and from one version of the platform to another. If you're not satisfied with the default button style and want to customize it to match the design of your application, then you can replace the button's background image with a state list drawable. A state list drawable is a drawable resource defined in XML that changes its image based on the current state of the button. Once you've defined a state list drawable in XML, you can apply it to your Button with the android:background attribute.

**public** **void** onClick(View arg0) {

// Clear the User session data

// and redirect user to LoginActivity

session.logoutUser();

}

});

btnCustomize.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

startActivity(**new** Intent(MainActivity.**this**, Customize.**class**));

}

});

btnPass.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

initiatePopupWindow();

}

});

}

***Method initiatePopupWindow-*** A popup window that can be used to display an arbitrary view. The popup window is a floating container that appears on top of the current activity. Popup window is like a dialog box that gains complete focus when it appears on screen. Like the activity popup also has its own GUI which the android developer can design. Popup appears in front of activity and gains focus. Popup are usually used to show some additional information or something user wants to know after an event takes place.

**private** **void** initiatePopupWindow() {

**try** {

// We need to get the instance of the LayoutInflater

LayoutInflater inflater = (LayoutInflater) MainActivity.**this**

.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);

View layout = inflater.inflate(R.layout.*pop\_up*,

(ViewGroup) findViewById(R.id.*lyt\_popUp*));

pwindo = **new** PopupWindow(layout, 700, 450, **true**);

pwindo.showAtLocation(layout, Gravity.*TOP*, 0, 200);

btnClosePopup = (Button) layout.findViewById(R.id.*btn\_close\_popup*);

btnClosePopup.setOnClickListener(cancel\_button\_click\_listener);

} **catch** (Exception e) {

e.printStackTrace();

}

}

**private** OnClickListener cancel\_button\_click\_listener = **new** OnClickListener() {

**public** **void** onClick(View v) {

pwindo.dismiss();

}

};

}

* UsserSessionManager– is the class when the user can access his account. Android Storage can be done in many ways and one of the way is shared preferences.Shared Preferences allows us to manage session.Since Session are useful when you want to store user data globally through out the application. This can be done in two ways. One is storing them in a global variables and second is storing the data in shared preferences

|  |  |
| --- | --- |
| Functions and constructor | Type and access level |
| createUserLoginSession | public void |
| checkLogin | boolean |
| logoutUser | public void |
| isUserLoggedIn | boolean |

Explications for Java codes

**import java.util.HashMap -** The **java::import** command provides a means to specify Java class names in a shortened format. This functionality is the equivalent of the *import* statement in Java. Once imported, the Java class**java.util.Hashtable** could be specified with the shortened name **Hashtable**. Each *class* argument is checked to ensure that it is the name of a accessible class and does not conflict with another class imported from another package. The optional **-package** *pkg* arguments are used to indicate the package that each *class* argument is imported from. If the **-package** *pkg* arguments are given, none of the*class* arguments can include a package specifier, meaning the '.' character must not appear in a *class* name.

***Method createUserLoginSession*** – Most applications need to know the identity of a user. Knowing a user's identity allows an app to provide a customized experience and grant them permissions to access their data. The process of proving a user's identity is called authentication. The code to authenticate a user varies by provider and transport method, but they all have similar signatures and accept a callback function. Use it to handle errors and process the results of a successful login.

**public** **void** createUserLoginSession(String name, String email){

// Storing login value as TRUE

editor.putBoolean(*IS\_USER\_LOGIN*, **true**);

// Storing name in pref

editor.putString(*KEY\_NAME*, name);

// Storing email in pref

editor.putString(*KEY\_EMAIL*, email);

// commit changes

editor.commit();

}

***Method checkLogin*** – Checks whether a cookie containing a token for the specified scope exists. Also, sets the current scope to the specified scope, so that subsequent calls to [logout()](https://developers.google.com/gdata/jsdoc/1.9/google/accounts/user#logout) and [getInfo()](https://developers.google.com/gdata/jsdoc/1.9/google/accounts/user" \l "getInfo) use the new current scope. The returned token is not guaranteed to be valid, so the application must be able to handle an authorization exception when the token is used.

**public** **boolean** checkLogin(){

// Check login status

**if**(!**this**.isUserLoggedIn()){

// user is not logged in redirect him to Login Activity

Intent i = **new** Intent(\_context, LoginActivity.**class**);

// Closing all the Activities from stack

i.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);

// Add new Flag to start new Activity

i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

// Staring Login Activity

\_context.startActivity(i);

**return** **true**;

}

**return** **false**;

}

***Method LogOutUser***– The Logout method clears all authentication cookies from the cookie cache and resets the static Thread. CurrentPrincipal property to a WindowsPrincipal object that contains the current WindowsIdentity. The SignOut method removes the forms-authentication ticket information from the cookie or the URL if CookiesSupported is false. You can use the SignOut method in conjunction with the RedirectToLoginPage method to log one user out and allow a different user to log in.

If you run exclusively in cookieless mode, or if you support both authenticated and anonymous users, you should explicitly control the redirect to the login page if you require special business logic to execute as a result of removing the anonymous identifier.

When the SignOut method is called, a redirect to the application's login page is made by calling the Redirect method with the endResponse parameter set to false. The redirect does not take place until the current page has finished executing, so additional code can be run. If the code does not contain an explicit redirect to another page, the user is redirected to the login page configured in the application's configuration file.

**public** **void** logoutUser(){

// Clearing all user data from Shared Preferences

editor.clear();

editor.commit();

// After logout redirect user to Login Activity

Intent i = **new** Intent(\_context, LoginActivity.**class**);

// Closing all the Activities

i.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);

// Add new Flag to start new Activity

i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

// Staring Login Activity

\_context.startActivity(i);

}

***Method isUserLoggedIn***– The SessionService allows access to the current sessions of the current context. The session objects that are cached by this service are obtained through a listener. The listener must be configured in your web.xml file.

**public** **boolean** isUserLoggedIn(){

**return** pref.getBoolean(*IS\_USER\_LOGIN*, **false**);

}

}

5. CONCLUSIONS

We choose to do this project because we find it the best way to make a service more efficient.

In this project, I weried to develop a log in App, during to develop an application . First we should choose which technology you want to use and which technology is more suitable for our application. For example we did this project by file, where we can save the informations.

Then analyze the program and design the classes and the relation between the classes, this step is most important because if you don’t prevision and don’t look-ahead or don’t pay attention to all facilities that the application can be have and make your project, probably during to develop the application you may encounter with the inspected problem. In this time, changing the structure needs more works and more time.

5. REFERENCES

<http://developer.android.com/training/basics/firstapp/building-ui.html>

<http://www.tutorialspoint.com/android/android_login_screen.htm>

https://developer.android.com/training/index.html