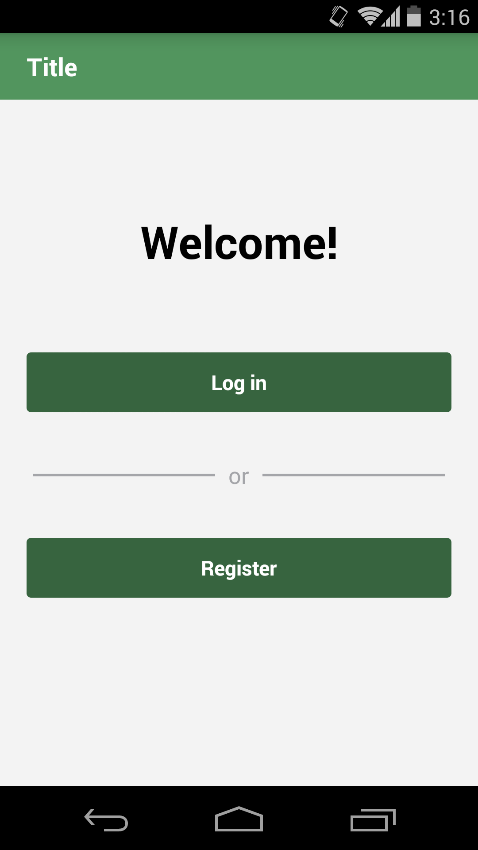
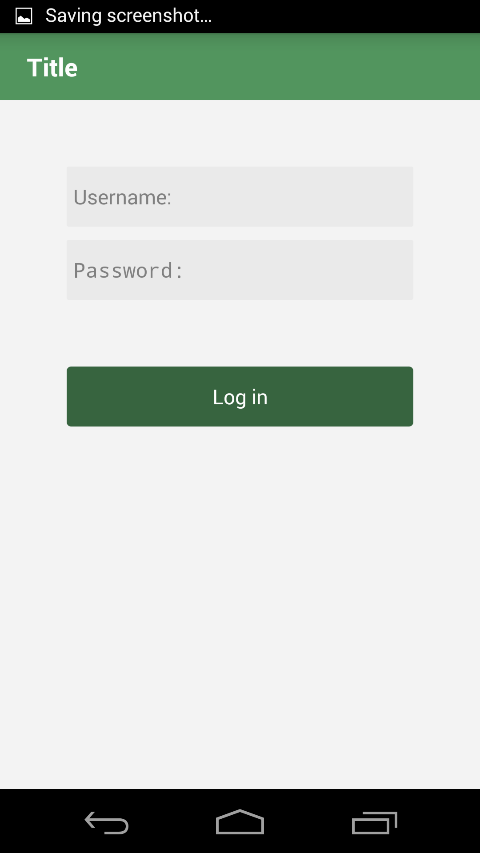
**Homework Documentation**

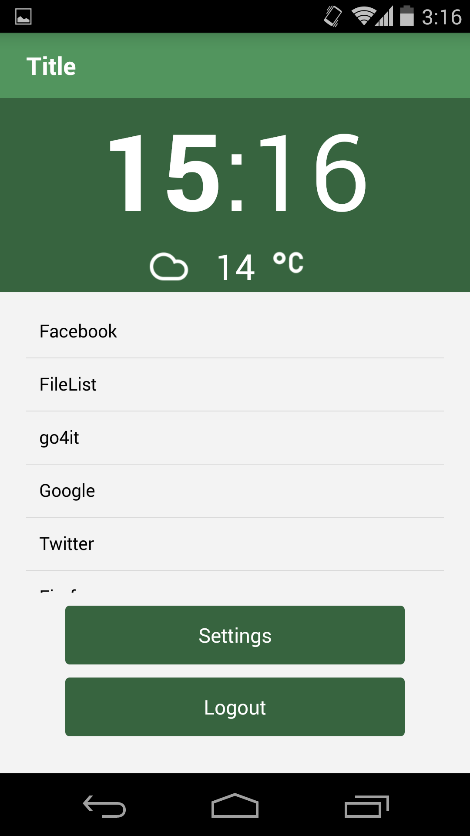
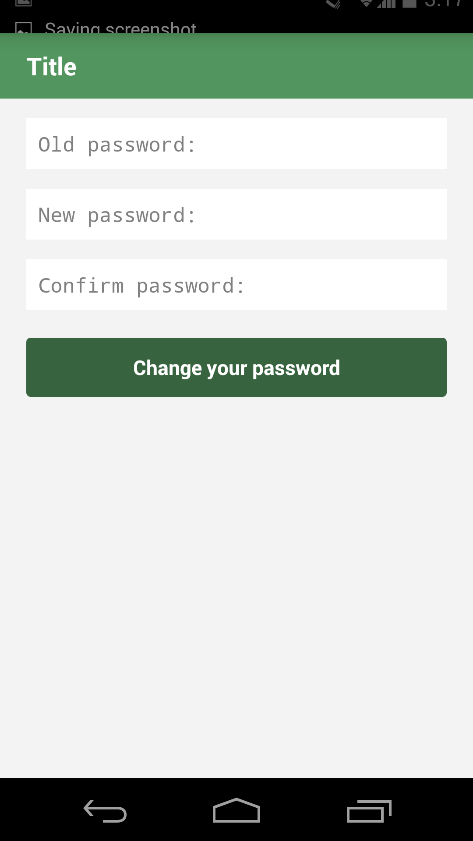
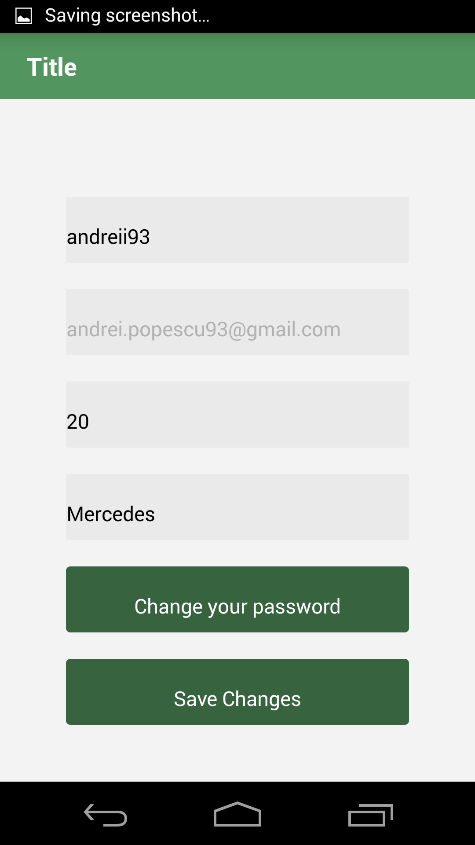
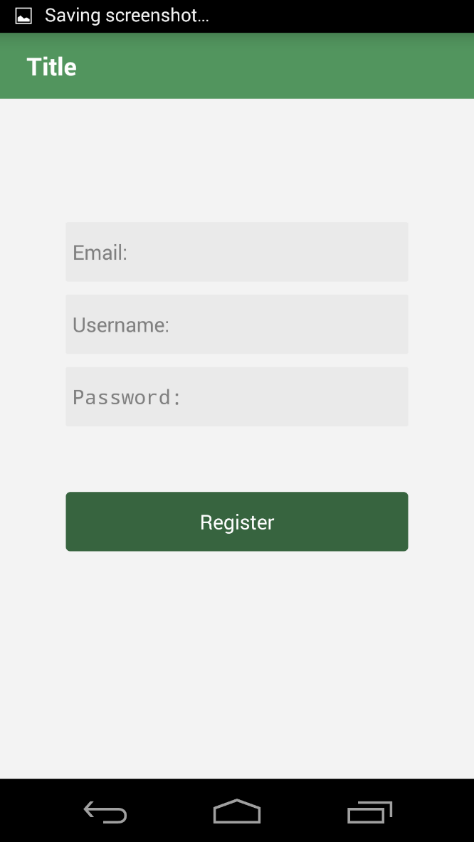
1. **Overview**
2. **Specifications**
3. **Prerequisites**
4. **User flow**
5. **Documentation**
6. **Overview**

The purpose of the project was to build an application for mobile devices. The user will be able to create and account and login in the app, where he will see the current time, current weather based on his current location and a list of links populated from a database. Moreover, the user can review his account settings and change them, and also his password.

The application is built on Android platform, the leading mobile operating system in the market with more than 70% market share.

The application was created using Android Development Tools (Eclipse IDE with the Android Software Development Kit plugin).

1. **Specifications**
   1. The application will display a Log In interface
   2. The user will be able to log in, or create an account
   3. As an authenticated user, one can create and modify the profile, with the data being saved in a database
   4. An authenticated user can change the password
   5. The user will be presented on the homepage the current weather for his current location.
   6. The user can access a list of hotlinks (populated from the database) and launch the specific websites directly from the application.
2. **Prerequisites**
   1. More than 85% of today Android users run at least version 4.0 (ICS) of the OS. Based on this, I developed the application for the minimum 4.0 SDK.
   2. The device must have an internet connection.
   3. The device must allow GPS Coarse Location for the weather to be retrieved.
   4. The application is targeting 4.4 Kit Kat OS.
   5. It uses Google Play Services for the Location APIs, so Google Play Services must be installed in order for the application to run (all devices back to Android 2.0 Éclair are using Google Play Services).
3. **User flow**



Login Screen

Home screen

Change Password Screen

Settings Screen

Register Screen

Welcome Screen

When launching the application, the user is presented with the Welcome Screen. Here, he can either login with an existing account using the “Login” Button or opt to create a new one via the “Register” Button.

Upon pressing the “Register” Button, the application will switch to the Register Screen where the user has to insert a desired username, an email and lastly a password. By pressing the register button in here, the user will be taken to the Login Screen, where he will type his username and password and will launch the Home screen.

On the Home screen, the user will notice in the upper part a Clock and right underneath it a field displaying the current temperature in Celsius degrees along with a picture representing the actual condition.

Underneath it, there is the list of links from the database. By clicking on the links, the application opens the webpage found at that link in the default browser.

Moreover, there are 2 buttons. One for the account settings called “Settings” and one for the logout.

Tapping on “Settings” will get the user to the Settings Screen where he can review the information, such as the username and also add new info (age & car). Tapping on “Save” will save the data in the database and display a message, after which the user can leave the screen. There is also a “Change Password” Button which advances the user to another screen, where he needs to input his old password and type a new one and also confirm it. After doing so and proceeding with changing the password, the user will be automatically logout and taken back to the login screen.

Finally, the logout button simply logs out the user and takes him back to the login screen.

1. **Documentation**

The application is structured in packages and uses a Model – View – Controller architecture.

The Model – View – Controller is a design pattern that separates the application in 3 parts.

1. The model is the structure used to store the needed information and to populate the view.
2. The view is the part that the user is seeing and is populated from the model with the controller’s help/
3. The controllers is used to obtain the input from the user’s interaction with the view, process it and finally store it in the model. It is also used to populate the view with the information found in the model.

The main package com.example.homework contains all the other packages and also some classes of its own.

The first subpackage is com.example.base . This contains the classes used by all the other classes in the application to extend from.

The BaseFragment, BaseFragmentActivity are used for defining what methods should the fragments and the activities in the app should have.

The BaseModel is used to extend all the models from it, so they can be passed as responses from the controllers (called APIs in the application) to the fragments and activities. ModelFailureResponse and ModelSuccessResponse are just classes extending BaseModel used to determine if an action completed successfully or not.

The interface BaseApiListener is used for providing the onResponse method in the activities and fragments. And the BaseApiInterface is used to set the ApiListener.

In the MainApp in Application used to connect to the Parse database.

The MainActivity is used just to provide the methods for displaying each fragment and set the initial fragment.

In com.example.homework.utils we find the PopDialog class used to provide methods for simply displaying dialogs in the app, and the ScreenListener interface used to change the Title bar’s title according to the displayed fragment.

In com.example.homework.weather we have the classes responsible for receiving the current condition. The OpenWeatherMapClient provides the method for getting the current condition with an AsyncTask in form of a JSON, the method for parsing that JSON Response and for specifying the condition type in the WeatherConditionModel. The WeatherDataSource is an interface for specifying the methods that are to be implemented and WeatherHelper is a class used to assign a drawable for each condition type.

In all the other packages, we find the fragments for each screen and the controllers and/or models responsible for the data in that fragments.

In com.example.homework. Home we also find a custom adapter, AdapterLinks. This is used for setting up the views in the ListView.