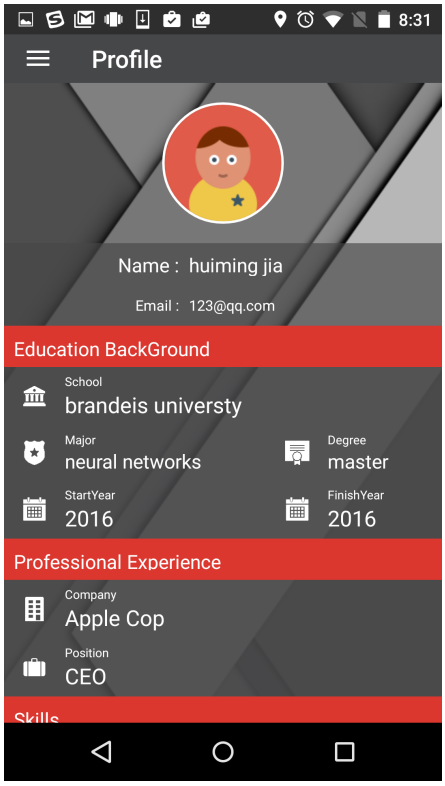
Resume Share Project Report

Huiming Jia, Hao Wang, Shuyi Lei, Chenyue Chen

December 02, 2016

Project Overview

Resume share is developed to help people easily share their own Resume or review others'.



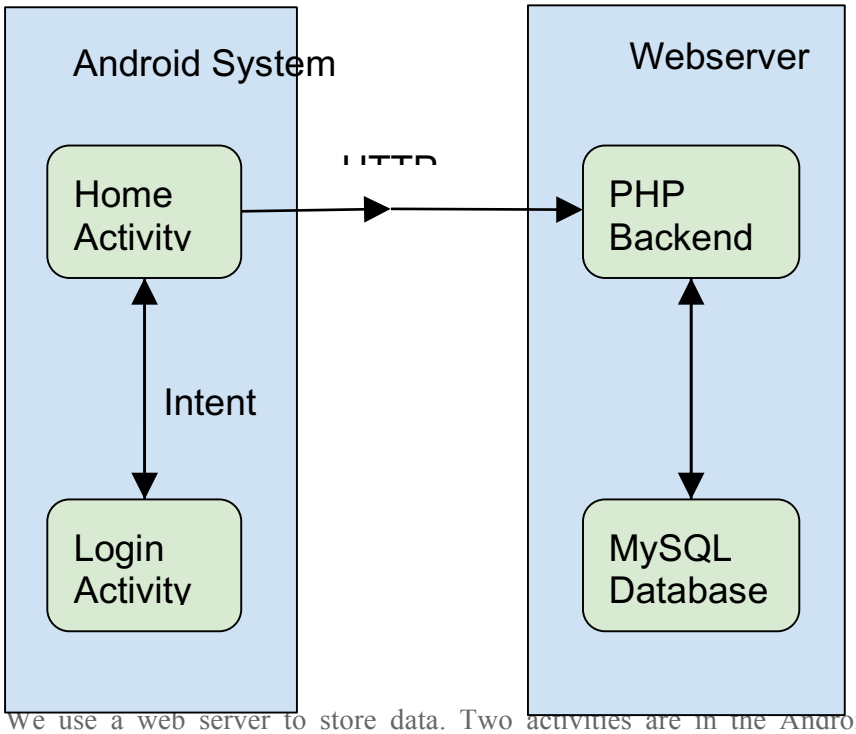
Problem we solve

Resume plays an important role in job hurting. In most cases, it decides whether an applicant get the interview opportunity. Traditionally, resumes are shared in paper or by email, both are not efficient enough. To solve this problem, we develop an application called resume share.

Design Pattern

Views, controller and models are three different levels of our Java classes. Views contains classes directly related to the UI. Controller maintain the main business logics, and plays an important role in functionalities. Models are directly related to data model, its for reading and writing data.

Communication and Data Flow



We use a web server to store data. Two activities are in the Android System. Between activities in Android System, data are passed by Intent. In our project, we need to synchronize our data with the server in order to interact with other users. That's the reason why we have a web server. To keep things simple, we use PHP programming language to build the server. MySQL database is used as data storage in the server.

HTTPTask

Web connection of Android System needs to be independent from Android UI, which means it runs asynchronously from the main thread, otherwise we will get a NetworkOnMainThreadException. It is always a bad idea to block Android main thread, which may interact with user and system anytime.

We use a basic abstract of asynchronized task in Android to handle HTTP requests. In the task, we send data as parameters in a GET request. We also define an interface called HttpTaskHandler, which have two

method, taskSuccessful and taskFailed, which provide the detail on the task finish, and will be required for different tasks.

HttpUploadImageTask and HttpLoadImageTask

This task is to upload image to the server. Since we are transporting binary files, we use these two classes to handle them. Image uploading can be directly handled by operating streams.

Some Android Library like Picasso provides some easy way to handle image related networking tasks. However, since we just need simple functions, we are able to implement it by ourselves.

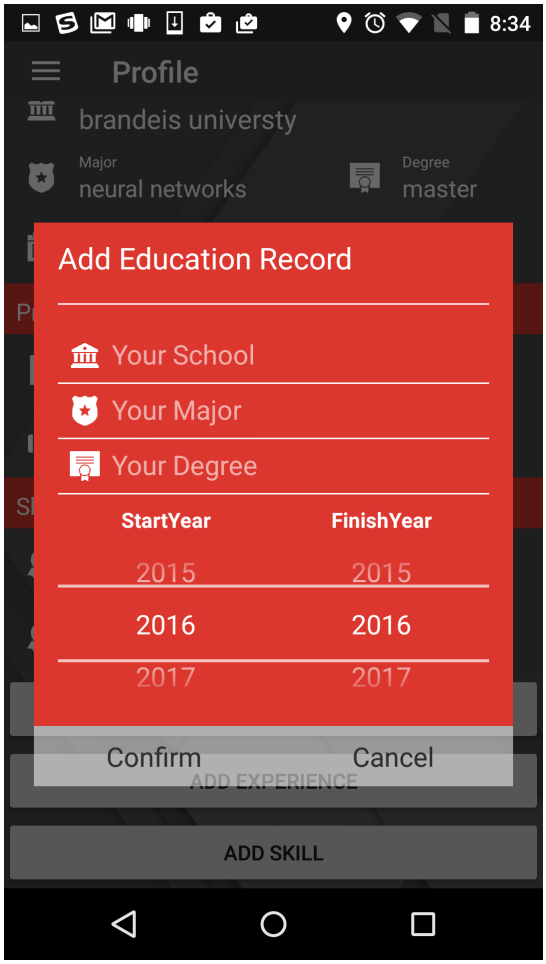
Functions and implementations

This section introduces some functions used in our project and the details about how we implement them.

Login and Register

The first activity for users when they open the app is a Login Activity. This activity is to provide the login and register transaction. The textview handles the switch from Login and Register transactions. Data is passed from activity to controller and from controller to model. Model executes http task. In the controller's login method, after the if statement checks to make sure the username and password are not empty and the network connection is successful, the loginfromRemote method from the User Model class is called. Similarly, the register method checks the consistency of the two sets of password as well as the correctness of the account format.

The addUserToRemote method creates a new user at the back end. Http task is an asynchronous task defined by us to set handler. Handler is a call back method. The task is executed after we finish setting the handler. The username and password are passed to the server by parameters set by us, and then a new user is registered. There are some other similar methods such as login method. Login method obtains the user's username and password and compares them with those in the database. If they match with each other, the JSON value is true. Besides this, the show information method uses the http task to get the user's personal information after the he/she inputs his/her username, and then the method put it into a class before it is passed to activity or fragment.



Base dialog and Resume Items

In the resume, different items are to be included. To add those items, several different dialogs are designed to handle them.

To keep a unique style and a modular design, we develop a BaseDialog class. Basedialog has three components: content, title and button (grey button). Since we add different content into the dialog class each time, we will have to define sub-class to accommodate those different contents. For example, since the user defines the number of buttons, the number will be determined by how many parameters imputed. The maximum number is three. All of the dialog extend the Basedialog class such as the Editdialog (its function is to edit the text box).

Find People Nearby

We assume the most cases of resume sharing are face to face, like in a job fair. The UI is designed to handle this requirement. When clicking the big red button, the set-on-touch-listener method is called. First it creates a wave effect, which represents the process of sending user's information to other people. It then creates a locationUtil object. This method gets called here is the key part of getting geographic info. The locationutil object will call getLongitude and getAltitude method to get its geographic information.

Besides, a time utility object is created to record the moment when the button is clicked. We send the account information, time, longitude and latitude to the controller.

In the location utility class, at first, a location manager object is initialized by passing the location service to get-system-service. Next, GetLastKnownLocation Method gets called to create a location object which includes the longitude and latitude information. Then the location information is updated through the updateView method. The updated information will then be returned by the getlongitude and getaltitude method that's in the home-fragment class.

After the information is sent to the controller, controller then sends the information to the instantLocationModel. What model class does here is basically communicating with the server. The add-instant-location-toRemote method here is responsible for that.

Four parameters are passed in the method. An asynchronous task is created to send request to the server. The server checks the distance between the newly added user with anyone clicked the button in 10 mins. When distance between two people is within 100 meters and the time period between two clicks is within 10 mins, the php code on the server side will determine whether it is a successful request or not. Then the result will be sent to both Android systems that show the paired partner's information, allowing them to send friend request to each other.

Summary and Future Improvement

The business logic behind the code works well in the situation we assume. The UI we implement is clean and efficient, and workflow of the project is complete, i.e., register, login, edit resume, find potential employers/employees nearby, and send/receive resumes from/to them. It especially suits for the circumstance where a Job Fair is hold and resume share is in a batch mode.

We plan to improve our project by subdividing the users into employees/employers. In this case, the "find people nearby" function will be more efficient, where employees/employers can only see each others'. Moreover, employers themselves don't have to have their own resume, and employees may submit resumes to employer remotely, by just search for their names.