```java
//author(s): Alex Berliner, Jimmie Potts, Alan Birmaher
package com.cyf.challengeyourfriends;

import java.util.ArrayList;
import java.util.List;

import twitter4j.auth.RequestToken;
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.app.Activity;
import android.app.LoaderManager.LoaderCallbacks;
import android.content.Context;
import android.content.CursorLoader;
import android.content.Intent;
import android.content.Loader;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

/**
 * A login screen that offers login via email/password.
 */
public class LoginActivity extends Activity implements LoaderCallbacks<Cursor> {

    void log(String str) {
        Log.w("Login", str);
    }

    /**
     * A dummy authentication store containing known user names and passwords.
     * TODO: remove after connecting to a real authentication system.
     */
    private static final String[] DUMMY_CREDENTIALS = new String[] {
            "foo@example.com:hello", "bar@example.com:world" };
    /**
     * Keep track of the login task to ensure we can cancel it if requested.
     */
    private UserLoginTask mAuthTask = null;
```

```
// UI references.
private AutoCompleteTextView mEmailView;
private EditText mPasswordView;
private View mProgressView;
private View mLoginFormView;

private boolean isUseStoredTokenKey = false;
private boolean isUseWebViewForAuthentication = false;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FriendManager f = new FriendManager(this);
    setContentView(R.layout.activity_login);

    Button mEmailSignInButton = (Button) findViewById(R.id.
        email_sign_in_button);
    mEmailSignInButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            attemptLogin();
        }
    });

    if(isUseStoredTokenKey)
        attemptLogin();
}

private void populateAutoComplete() {
    getLoaderManager().initLoader(0, null, this);
}

/**
 * Attempts to sign in or register the account specified by the login form.
 * If there are form errors (invalid email, missing fields, etc.), the
 * errors are presented and no actual login attempt is made.
 */
//returns based on the status of the network
boolean checkCon() {
    ConnectivityManager connectivity = (ConnectivityManager) getSystemService
        (Context.CONNECTIVITY_SERVICE);
    if (connectivity != null) {
        NetworkInfo[] info = connectivity.getAllNetworkInfo();
        if (info != null)
            for (int i = 0; i < info.length; i++)
                if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                    return true;
                }

    }
    return false;
}
//attempts to open twitter with the authentication key
public void attemptLogin() {
    SharedPreferences sharedPreferences = PreferenceManager.
```

```
            getDefaultSharedPreferences(
                getApplicationContext());
        if(!sharedPreferences.getBoolean(ConstantValues.
            PREFERENCE_TWITTER_IS_LOGGED_IN, false)){
            new TwitterAuthenticateTask().execute();
        }
        else {
            Intent intent = new Intent(LoginActivity.this, TwitterActivity.class)
                ;
            startActivity(intent);
        }

        /*
        log("connection enabled: " + checkCon());
        // IMPORTANT REENABLE LATER
         Intent intent = new Intent(this, MenuActivity.class);
         startActivity(intent);

         * if (mAuthTask != null) { return; }
         *
         * // Reset errors. mEmailView.setError(null);
         * mPasswordView.setError(null);
         *
         * // Store values at the time of the login attempt. String email =
         * mEmailView.getText().toString(); String password =
         * mPasswordView.getText().toString();
         *
         * boolean cancel = false; View focusView = null;
         *
         * // Check for a valid password, if the user entered one. if
         * (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
         * mPasswordView.setError(getString(R.string.error_invalid_password));
         * focusView = mPasswordView; cancel = true; }
         *
         * // Check for a valid email address. if (TextUtils.isEmpty(email)) {
         * mEmailView.setError(getString(R.string.error_field_required));
         * focusView = mEmailView; cancel = true; } else if
         * (!isEmailValid(email)) {
         * mEmailView.setError(getString(R.string.error_invalid_email));
         * focusView = mEmailView; cancel = true; }
         *
         * if (cancel) { // There was an error; don't attempt login and focus
         * the first // form field with an error. focusView.requestFocus(); }
         * else { // Show a progress spinner, and kick off a background task to
         * // perform the user login attempt. showProgress(true); mAuthTask =
         * new UserLoginTask(email, password); mAuthTask.execute((Void) null); }
         */
    }


    class TwitterAuthenticateTask extends AsyncTask<String, String, RequestToken>
        {

        @Override
        protected void onPostExecute(RequestToken requestToken) {
```

```
            Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(requestToken
                .getAuthenticationURL()));
            startActivity(intent);
        }

        @Override
        protected RequestToken doInBackground(String... params) {
            // TODO Auto-generated method stub
            return TwitterUtil.getInstance().getRequestToken();
        }
    }




    private boolean isEmailValid(String email) {
        // TODO: Replace this with your own logic
        return email.contains("@");
    }

    private boolean isPasswordValid(String password) {
        // TODO: Replace this with your own logic
        return password.length() > 4;
    }

    /**
     * Shows the progress UI and hides the login form.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    public void showProgress(final boolean show) {
        // On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
        // for very easy animations. If available, use these APIs to fade-in
        // the progress spinner.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
            int shortAnimTime = getResources().getInteger(
                    android.R.integer.config_shortAnimTime);

            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
            mLoginFormView.animate().setDuration(shortAnimTime)
                    .alpha(show ? 0 : 1)
                    .setListener(new AnimatorListenerAdapter() {
                        @Override
                        public void onAnimationEnd(Animator animation) {
                            mLoginFormView.setVisibility(show ? View.GONE
                                    : View.VISIBLE);
                        }
                    });

            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mProgressView.animate().setDuration(shortAnimTime)
                    .alpha(show ? 1 : 0)
                    .setListener(new AnimatorListenerAdapter() {
                        @Override
                        public void onAnimationEnd(Animator animation) {
                            mProgressView.setVisibility(show ? View.VISIBLE
                                    : View.GONE);
```

```
                    }
                });
        } else {
            // The ViewPropertyAnimator APIs are not available, so simply show
            // and hide the relevant UI components.
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
        }
    }

    @Override
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this,
                // Retrieve data rows for the device user's 'profile' contact.
                Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
                        ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
                ProfileQuery.PROJECTION,

                // Select only email addresses.
                ContactsContract.Contacts.Data.MIMETYPE + " = ?",
                new String[] { ContactsContract.CommonDataKinds.Email.
                    CONTENT_ITEM_TYPE },

                // Show primary email addresses first. Note that there won't be
                // a primary email address if the user hasn't specified one.
                ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");
    }

    @Override
    public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
        List<String> emails = new ArrayList<String>();
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            emails.add(cursor.getString(ProfileQuery.ADDRESS));
            cursor.moveToNext();
        }

        addEmailsToAutoComplete(emails);
    }

    @Override
    public void onLoaderReset(Loader<Cursor> cursorLoader) {

    }

    private interface ProfileQuery {
        String[] PROJECTION = { ContactsContract.CommonDataKinds.Email.ADDRESS,
                ContactsContract.CommonDataKinds.Email.IS_PRIMARY, };

        int ADDRESS = 0;
        int IS_PRIMARY = 1;
    }

    private void addEmailsToAutoComplete(List<String> emailAddressCollection) {
        // Create adapter to tell the AutoCompleteTextView what to show in its
        // dropdown list.
```

```
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                LoginActivity.this,
                android.R.layout.simple_dropdown_item_1line,
                emailAddressCollection);

        mEmailView.setAdapter(adapter);
    }

    /**
     * Represents an asynchronous login/registration task used to authenticate
     * the user.
     */
    public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

        private final String mEmail;
        private final String mPassword;

        UserLoginTask(String email, String password) {
            mEmail = email;
            mPassword = password;
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            // TODO: attempt authentication against a network service.

            try {
                // Simulate network access.
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                return false;
            }

            for (String credential : DUMMY_CREDENTIALS) {
                String[] pieces = credential.split(":");
                if (pieces[0].equals(mEmail)) {
                    // Account exists, return true if the password matches.
                    return pieces[1].equals(mPassword);
                }
            }

            // TODO: register the new account here.
            return true;
        }

        @Override
        protected void onPostExecute(final Boolean success) {
            mAuthTask = null;
            showProgress(false);

            if (success) {
                finish();
            } else {
                mPasswordView
                        .setError(getString(R.string.error_incorrect_password));
                mPasswordView.requestFocus();
```

```
            }
        }

        @Override
        protected void onCancelled() {
            mAuthTask = null;
            showProgress(false);
        }
    }
}
```