```java
/* Help.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This activity displays all of the help information using a
   recycler view.
*/
package com.fbla.dulaney.fblayardsale;

import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.view.View;

import com.fbla.dulaney.fblayardsale.databinding.ActivityHelpBinding;

public class Help extends AppCompatActivity implements View.OnClickListener
{
    ActivityHelpBinding mBinding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_help);

        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_help);
        mBinding.done.setOnClickListener(this);
        mBinding.listHelp.setLayoutManager(new LinearLayoutManager(this));
        mBinding.listHelp.setAdapter(new HelpAdapter());
        setSupportActionBar(mBinding.myToolbar);
    }

    @Override
    public void onClick(View v)
    {
        switch(v.getId())
        {
            case R.id.done:
                this.finish();
                break;
        }
    }

    @Override
    public void onBackPressed()
    {
        this.finish();
    }
}

```

```java
/* MySales.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This activity lists all of the items you have for sale. It allows
   you to delete any item, or look at their comments.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import com.fbla.dulaney.fblayardsale.controller.MySalesController;
import com.fbla.dulaney.fblayardsale.databinding.ActivityMysalesBinding;

public class MySales extends AppCompatActivity implements View.OnClickListener, FblaAzure.
LogonResultListener {
    private ActivityMysalesBinding mBinding;
    private FblaAzure mAzure;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mysales);
        Bundle b = getIntent().getExtras();
        String userId = b.getString("userId");
        String token = b.getString("token");
        if (userId == null || token == null) {
            Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
            finish();
            return;
        }

        mAzure = new FblaAzure(this);
        mAzure.setLogonListener(this);
        mAzure.doLogon(userId, token);

        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_mysales);
        mBinding.list.setLayoutManager(new LinearLayoutManager(this));
        setSupportActionBar(mBinding.myToolbar);

        Log.d("MySales", "onCreate");
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {

            case R.id.comments:
                Intent i = new Intent(this, Comments.class);
                Bundle b = new Bundle();
                b.putString("userId", mAzure.getUserId());
                b.putString("token", mAzure.getToken());
                i.putExtras(b);
                this.startActivity(i);
                break;
            default:
                break;
        }
    }

    @Override
    public void onBackPressed()
    {
        this.finish();
    }

    @Override
    public void onLogonComplete(Exception e) {
        MySalesAdapter adapter = new MySalesAdapter(this, this, mAzure);
        MySalesController.AttachAdapter(adapter);
        mBinding.list.setAdapter(adapter);
    }
}

```

```java
/* AddSales.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This activity is used to add a new sale item.

   Both Name and Price are required for an item. This is enforced by only enabling
   the Save button when both have data.

   This activity also interacts with the phone's photo gallery and camera in order
   to include a picture of the item.

   Pictures are saved to Azure storage using FblaPicture.
*/
package com.fbla.dulaney.fblayardsale;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.databinding.DataBindingUtil;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import com.fbla.dulaney.fblayardsale.controller.MySalesController;
import com.fbla.dulaney.fblayardsale.databinding.ActivityAddsalesBinding;

import java.io.InputStream;
import java.util.UUID;

import com.fbla.dulaney.fblayardsale.model.*;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

public class AddSales extends AppCompatActivity implements View.OnClickListener, FblaAzure.
LogonResultListener {
    private ActivityAddsalesBinding mBinding;
    private MobileServiceTable<SaleItem> mSaleItemTable;
    private FblaAzure mAzure;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_addsales);
        Bundle b = getIntent().getExtras();
        String userId = b.getString("userId");
        String token = b.getString("token");
        if (userId == null || token == null) {
            Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
            finish();
            return;
        }

        mAzure = new FblaAzure(this);
        mAzure.setLogonListener(this);
        mAzure.doLogon(userId, token);

        mSaleItemTable = mAzure.getClient().getTable(SaleItem.class);

        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_addsales);
        FblaPicture.setLayoutImage(mBinding.activityAddsales);
        setSupportActionBar(mBinding.myToolbar);
        mBinding.gallery.setOnClickListener(this);
        mBinding.camera.setOnClickListener(this);
        mBinding.back.setOnClickListener(this);
        mBinding.finish.setOnClickListener(this);
        mBinding.another.setOnClickListener(this);

        mBinding.finish.setEnabled(false);
        mBinding.another.setEnabled(false);

        // Make sure Name is required.
        mBinding.editname.addTextChangedListener(new TextWatcher() {
            public void onTextChanged(CharSequence s, int start, int before, int count) {

            }

            public void beforeTextChanged(CharSequence s, int start, int count, int after)
{

            }

            public void afterTextChanged(Editable s) {
                if (s.length() > 0 && mBinding.editprice.getText().length() > 0) {
                    mBinding.finish.setEnabled(true);
                    mBinding.another.setEnabled(true);
                } else {
                    mBinding.finish.setEnabled(false);
                    mBinding.another.setEnabled(false);
                }
            }
        });

        // Make sure Price is required.
        mBinding.editprice.addTextChangedListener(new TextWatcher() {
            public void onTextChanged(CharSequence s, int start, int before, int count) {

            }

            public void beforeTextChanged(CharSequence s, int start, int count, int after)
{

            }

            public void afterTextChanged(Editable s) {
                if (s.length() > 0 && mBinding.editname.getText().length() > 0) {
                    mBinding.finish.setEnabled(true);
                    mBinding.another.setEnabled(true);
                } else {
                    mBinding.finish.setEnabled(false);
                    mBinding.another.setEnabled(false);
                }
            }
        });
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.gallery:
                // Load a picture from the phone's gallery
                // Ask for permission first
                if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE);
                    if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
                        // Should we show an explanation?
                        if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.READ_EXTERNAL_STORAGE)) {
                            // Explain to the user why we need to read the contacts
                        } else {
                            ActivityCompat.requestPermissions(this,
                                    new String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, 0);
                        }
                        return;
                    }
                }

                Intent i = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                Log.d("CameraFragment", "Starting GALLERY Intent");
                this.startActivityForResult(i, 1);
                break;
            case R.id.camera:
                // Take a picture from the camera.
                // Ask for permission first
                if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA);
                    if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
                        // Should we show an explanation?
                        if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA)) {
                            // Explain to the user why we need to read the contacts
                        } else {
                            ActivityCompat.requestPermissions(this,
                                    new String[]{Manifest.permission.CAMERA}, 0);
                        }
                        return;
                    }
                }

                Intent j = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                this.startActivityForResult(j, 2);
                break;
            case R.id.another:
                // Save the current item, and reload the activity to be ready for another one.
                addItem(v);
                this.finish();
                Intent it = new Intent(this, AddSales.class);
                Bundle b = new Bundle();
                b.putString("userId", mAzure.getUserId());
                b.putString("token", mAzure.getToken());
                it.putExtras(b);
                this.startActivity(it);
                break;
            case R.id.finish:
                // Save the current item and return to YardSaleMain.
                addItem(v);
                this.finish();
                break;
            default:
                // Don't save anything, just return to YardSaleMain.
                this.finish();
                break;
        }
    }

    @Override
    public void onBackPressed()
    {
        this.finish();
    }

    // Add a new item to the database.
    private void addItem(View view) {
        if (!mAzure.getLoggedOn()) return;

        // Create a new item from the SaleItem model.
        final SaleItem item = new SaleItem();
        item.setId(UUID.randomUUID().toString());
        item.setName(mBinding.editname.getText().toString());
        item.setAccount(mAzure.getAccount());
        item.setDescription(mBinding.editdesc.getText().toString());
        String sPrice = mBinding.editprice.getText().toString();
        if (sPrice == null || sPrice.equals("")) item.setPrice(0);
        else item.setPrice(Float.parseFloat(mBinding.editprice.getText().toString()));
        Bitmap b = FblaPicture.GetPictureFromView(mBinding.picture);
        if (b != null) {
            item.setPicture(b);
        }
        MySalesController.addItem(item);

        // Save the item to the database over the internet.
        AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
            @Override
            protected Void doInBackground(Void... params) {
                try {
                    mSaleItemTable.insert(item);
                    Log.d("AddSales:insert", "Created item " + item.getName());
                    if (item.getHasPicture()) {
                        Bitmap picture = item.getPicture();
                        FblaPicture.UploadImage(item.getId(), picture);
                        // For some strange reason, uploading the picture destroys it on the
  item. So put it back.
                        item.setPicture(picture);
                    }
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                        }
                    });
                } catch (Exception e) {
                    Log.d("AddSales:insert", e.toString());
                }
                return null;
            }
        };
        task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        // Results can come from the Camera, Gallery, or the Comments activity.
        if (resultCode == android.app.Activity.RESULT_OK) {
            if (requestCode == 2 && data != null) { // From Camera
                Log.d("AddSales", "Result from Camera");

                try {
                    Bundle extras = data.getExtras();
                    Bitmap image = (Bitmap) extras.get("data");
                    image = FblaPicture.ResizePicture(this.getApplicationContext(), image);
                    FblaPicture.LoadPictureOnView(mBinding.picture, image);
                } catch (Exception ex) {
                    Log.e("AddSales:camera", ex.getMessage());
                }
            } else if (requestCode == 1 && data != null) // Gallery
            {
                // Gallery
                Log.d("AddSales", "Result from Gallery");

                try {
                    Uri pickedImage = data.getData();
                    InputStream stream = getContentResolver().openInputStream(pickedImage);
                    Bitmap image = BitmapFactory.decodeStream(stream);
                    image = FblaPicture.ResizePicture(this.getApplicationContext(), image);
                    FblaPicture.LoadPictureOnView(mBinding.picture, image);
                } catch (Exception ex) {
                    Log.e("AddSales:gallery", ex.getMessage());
                }
            }
        }
    } // onActivityResult

    @Override
    public void onLogonComplete(Exception e) {

    }
}
```

```java
/* Comments.java
   =============================================================================
                        Josh Talley and Daniel O'Donnell
                               Dulaney High School
                     Mobile Application Development 2016-17
   =============================================================================
   Purpose: This activity allows you to both add new comments, review existing
   comments posted on a sale item, and delete comments.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Context;
import android.databinding.DataBindingUtil;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Toast;

import com.fbla.dulaney.fblayardsale.controller.CommentListController;
import com.fbla.dulaney.fblayardsale.databinding.ActivityCommentsBinding;
import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.ItemComment;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

import java.util.UUID;

public class Comments extends AppCompatActivity implements View.OnClickListener, FblaAzure.
LogonResultListener {
    private ActivityCommentsBinding mBinding;
    private MobileServiceTable<ItemComment> mCommentTable;
    private FblaAzure mAzure;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_comments);
        Bundle b = getIntent().getExtras();
        String userId = b.getString("userId");
        String token = b.getString("token");
        if (userId == null || token == null) {
            Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
            finish();
            return;
        }

        mAzure = new FblaAzure(this);
        mAzure.setLogonListener(this);
        mAzure.doLogon(userId, token);

        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_comments);
        mBinding.post.setOnClickListener(this);
        mBinding.list.setLayoutManager(new LinearLayoutManager(this));
        setSupportActionBar(mBinding.myToolbar);

        mCommentTable = mAzure.getClient().getTable(ItemComment.class);

        Log.d("Comments", "onCreate");
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.post:
                if (!mBinding.newcomment.getText().toString().equals("")) {
                    addItem(v);
                }
                break;
            default:
                break;
        }
    }

    @Override
    public void onBackPressed()
    {
        this.finish();
    }

    // Add a new item to the database.
    private void addItem(View view) {
        if (!mAzure.getLoggedOn()) return;

        // Create a new comment from the ItemComment model.
        final ItemComment comment = new ItemComment();
        comment.setId(UUID.randomUUID().toString());
        comment.setComment(mBinding.newcomment.getText().toString());
        comment.setUserId(mAzure.getUserId());
        comment.setItemId(CommentListController.getItem().getId());
        comment.setAccount(mAzure.getAccount());
        CommentListController.addComment(comment);

        // Save the item to the database over the internet.
        AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
            @Override
            protected Void doInBackground(Void... params) {
                try {
                    mCommentTable.insert(comment);
                    Log.d("Comments:insert", "Created comment " + comment.getComment());
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                        }
                    });
                } catch (Exception e) {
                    Log.d("Comments:insert", e.toString());
                }
                return null;
            }
        };
        task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
        if (this.getCurrentFocus() != null) {
            InputMethodManager imm = (InputMethodManager) getSystemService(Context.
INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(this.getCurrentFocus().getWindowToken(), 0);
            mBinding.newcomment.setText("");
        }
    }

    @Override
    public void onLogonComplete(Exception e) {
        CommentsAdapter adapter = new CommentsAdapter(this, this, mAzure);
        CommentListController.AttachAdapter(adapter);
        mBinding.list.setAdapter(adapter);
    }
}

```

```java
/* FblaAzure.java
   =============================================================================
                            Josh Talley and Daniel O'Donnell
                                   Dulaney High School
                         Mobile Application Development 2016-17
   =============================================================================
   Purpose: This class establishes the connection with the Azure Mobile App server
   and the entire logon process. Part of the logon includes creating and/or fetching
   the user's Account information. It's done with the logon to make sure communication
   with the Azure server is working. This class extends AsyncTask because almost all
   of the login processes must be done in the background.

   Users must create or use a Microsoft account. The only piece of information about the
   account that is stored in the database is Azure's user token string, which
   looks like this: sid:fb96bd335c1fba115e191a4526df5353
   Also, when using the Microsoft provider, we have to continually clear cookies
   because the token caching interferes with our ability to logoff.

   The first time a user logs in, a new row is inserted into the Account table.
   The user's Account object and corresponding Schools object are stored as static
   variables in this class so that they are available to all activities and fragments
   in this mobile app. In addition, this class also stores the client object for
   Azure as a static variable, also to make is easily available to all activities
   and fragments in this mobile app.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.util.Log;
import android.webkit.CookieManager;
import android.webkit.ValueCallback;

import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.Schools;
import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;
import com.microsoft.windowsazure.mobileservices.MobileServiceClient;
import com.microsoft.windowsazure.mobileservices.MobileServiceException;
import com.microsoft.windowsazure.mobileservices.authentication.
MobileServiceAuthenticationProvider;
import com.microsoft.windowsazure.mobileservices.authentication.MobileServiceUser;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

public class FblaAzure {
    final private static String AZUREURL = "https://fbla-yardsale.azurewebsites.net";
    // Setup to use either Google+ or Microsoft.
    // However, Azure was updated and suddenly the Google logon doesn't work anymore.
    // Therefore, we will use Microsoft Accounts to authenticate.
    final private static MobileServiceAuthenticationProvider PROVIDER =
MobileServiceAuthenticationProvider.MicrosoftAccount;

    private MobileServiceUser mUser = null;
    private MobileServiceClient mClient = null;
    private Account mAccount = null;
    private MobileServiceTable<Account> mAccountTable = null;
    private MobileServiceTable<Schools> mSchoolsTable = null;

    private Context mContext;
    private ArrayList<LogonResultListener> mListeners = new ArrayList<LogonResultListener>(
);

    public FblaAzure (Context context) {
        mContext = context;
        try {
            mClient = new MobileServiceClient(AZUREURL, mContext);
        } catch (Exception e) {
            Log.d("FblaAzure:init", e.toString());
            mClient = null;
        }
    }

    // It seems to use WebKit to perform the OAuth authentication via Azure.
    // If successful, load the Account.  Otherwise return an exception to notify
    // the listeners that it didn't work.
    public void doLogon() {
        Log.d("FblaAzure", "Logging on...");
        ListenableFuture<MobileServiceUser> futureUser = mClient.login(PROVIDER);
        Futures.addCallback(futureUser, new FutureCallback<MobileServiceUser>() {
            @Override
            public void onFailure(Throwable exc) {
                onLogonFailure((Exception)exc);
            }
            @Override
            public void onSuccess(MobileServiceUser user) {
                mUser = user;
                doLoadAccount();
            }
        });
    }

    // This will load the account in the background.
    public void doLoadAccount() {
        new AsyncTask<Object, Object, Object>() {
            @Override
            protected Object doInBackground(Object... params) {
                Log.d("FblaAzure", "Loading Account...");
                return loadAccount();
            }
            @Override
            protected void onPostExecute(Object result) {
                if (result == null) onLogonSuccess();
                else onLogonFailure((Exception)result);
            }
        }.execute();
    }
    public void doLogon(String userId, String token) {
        mUser = new MobileServiceUser(userId);
        mUser.setAuthenticationToken(token);
        mClient.setCurrentUser(mUser);

        new AsyncTask<Object, Object, Object>() {
            @Override
            protected Object doInBackground(Object... params) {
                Log.d("FblaAzure", "Loading Account...");
                return loadAccount();
            }
            @Override
            protected void onPostExecute(Object result) {
                if (result == null) onLogonSuccess();
                else onLogonFailure((Exception)result);
```

```java
            }
        }.execute();
    }

    public void doLogoff(YardSaleMain main) {
        clearCookies();
        mAccountTable = null;
        mAccount = null;
        mUser = null;
        final YardSaleMain m = main;
        ListenableFuture<MobileServiceUser> mLogout = mClient.logout();
        Futures.addCallback(mLogout, new FutureCallback<MobileServiceUser>() {
            @Override
            public void onFailure(Throwable exc) {

            }
            @Override
            public void onSuccess(MobileServiceUser user) {
                mClient = null;
                m.finish();
            }
        });
        Log.d("FblaAzure:Logoff", "Logged Off");
    }

    public boolean getLoggedOn() { return mUser != null;}

    public MobileServiceClient getClient() {
        return mClient;
    }

    public String getUserId() {
        if (mUser == null) return null;
        else return mUser.getUserId();
    }
    public String getToken() {
        if (mUser == null) return null;
        else return mUser.getAuthenticationToken();
    }

    public Account getAccount() {
        return mAccount;
    }
    public void setAccount(Account account) { mAccount = account; }

    public int getSearchMiles(Context context) {
        if (context == null) return 5;
        SharedPreferences prefs = context.getSharedPreferences("settings", Context.
MODE_PRIVATE);
        if (prefs == null) return 5;
        return prefs.getInt("miles", 5);
    }

    public void setSearchMiles(Context context, int miles) {
        SharedPreferences prefs = context.getSharedPreferences("settings", Context.
MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putInt("miles", miles);
        editor.commit();
    }

    // Once all of the logon processes are complete, notify any listeners
    private void onLogonSuccess() {
        Log.d("FblaAzure", "onLogonSuccess");
        for (LogonResultListener listener : mListeners) {
            listener.onLogonComplete(null);
        }
    }

    // Notify any listeners that the logon has failed.
    private void onLogonFailure(Exception e) {
        for (LogonResultListener listener : mListeners) {
            listener.onLogonComplete(e);
        }
        Log.d("FblaAzure:Failure", e.toString());
    }

    // It seems to use WebKit to perform the OAuth authentication via Azure.
    // If successful, load the Account.  Otherwise return an exception to notify
    // the listeners that it didn't work.
    private Object providerLogon() {
        try {
            Log.d("FblaAzure:login", "Logging on to provider");
            mUser = mClient.login(PROVIDER).get();
            Log.d("FblaAzure:login", "Logged On");
            return loadAccount();
        } catch (Exception ex) {
            Log.d("FblaAzure:login", ex.toString());
            return ex;
        }
    }

    // A successfully loaded account means the token actually works and we can talk to Azure.
    // Some accounts may not be linked to a School.
    private Object loadAccount() {
        // Now load the account
        mAccountTable = mClient.getTable(Account.class);
        try {
            mAccount = mAccountTable.lookUp(mUser.getUserId()).get();
            // Found the account record, so set it on the Data object.
            Log.d("FblaAzure:account", "onSuccess - " + mAccount.getId());
            return loadSchool();
        } catch (ExecutionException e) {
            if (e.getCause().getClass() == MobileServiceException.class) {
                MobileServiceException mEx = (MobileServiceException) e.getCause();
                if (mEx.getResponse() != null && mEx.getResponse().getStatus().code == 404) { // Not Found
                    // The user is not in the table, so insert a new record for them.
                    mAccount = new Account();
                    mAccount.setId(mUser.getUserId());
                    mAccountTable.insert(mAccount);
                    Log.d("FblaAzure:account", "AccountEdit Created");
                    return null;
                } else {
                    Log.d("FblaAzure:account", mEx.toString());
                    return mEx;
                }
            } else {
                Log.d("FblaAzure:account", e.toString());
                return e;
            }
        } catch (Exception ex) {
            // Something else bad happened.
            Log.d("FblaAzure:account", ex.toString());
            return ex;
        }
    }

    // Load the school, if the Account is linked to one.
    // Return a null if everything is successful. Otherwise return an Exception.
    private Object loadSchool() {
        if (mAccount.getSchoolId() != null && mAccount.getSchool() == null) {
            Log.d("FblaAzure:school", "School " + mAccount.getSchoolId());
            mSchoolsTable = mClient.getTable(Schools.class);
            try {
                Schools school = mSchoolsTable.lookUp(mAccount.getSchoolId()).get();
                // Found the account record, so set it on the Data object.
                Log.d("FblaAzure:school", "onSuccess - "+school.getId());
                mAccount.setSchool(school);
                return null;
            } catch (ExecutionException e) {
                if (e.getCause().getClass() == MobileServiceException.class) {
                    MobileServiceException mEx = (MobileServiceException) e.getCause();
                    if (mEx.getResponse() != null && mEx.getResponse().getStatus().code == 404) { // Not Found
                        Log.d("FblaAzure:school", "School Missing");
                        return null;
                    } else {
                        Log.d("FblaAzure:school", mEx.toString());
                        return mEx;
                    }
                } else {
                    Log.d("FblaAzure:school", e.toString());
                    return e;
                }
            } catch (Exception ex) {
                // Something else bad happened.
                Log.d("FblaAzure:school", ex.toString());
                return ex;
            }
        } else {
            Log.d("FblaAzure:school", "No School");
            return null;
        }
    }

    private void clearCookies() {
        // Clear cookies and cache in order to logoff
        CookieManager.getInstance().removeAllCookies(new ValueCallback<Boolean>() {
            @Override
            public void onReceiveValue(Boolean value) {
                Log.d("FblaAzure:cookies", "Cookies cleared");
            }
        });
    }

    // Add a listener to call after logon is complete
    public void setLogonListener(LogonResultListener listener) {
        mListeners.add(listener);
    }

    // This is the interface to use on the logon callbacks.
    interface LogonResultListener {
        void onLogonComplete(Exception e);
    }
}

```

```java
/* AccountEdit.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This activity is used to display and edit account information. When
   a user first logs it, you are forwarded directly to this activity.

   The user's name is forced to be required by only enabling the save button
   when you put type something in the name field.

   You can search for a school using either just a zip code, or by selecting
   a state and type in a city. The city search is done using a "starts with"
   search, so you do not have to type in the whole name. You can change the
   school at any time. Doing so will "move" all of your items to that new school.
   You may also see a different set of "nearby" or local schools, in the local
   tab or on the map, who have items for sale.

   You can also change the search radius for schools in your local area, between
   either 5 miles or 10 miles.
*/
package com.fbla.dulaney.fblayardsale;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Toast;

import com.fbla.dulaney.fblayardsale.controller.LocalController;
import com.fbla.dulaney.fblayardsale.databinding.ActivityAccountBinding;
import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.Schools;
import com.fbla.dulaney.fblayardsale.model.ZipCodes;
import com.microsoft.windowsazure.mobileservices.MobileServiceList;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
import com.microsoft.windowsazure.mobileservices.table.query.QueryOrder;

import java.util.ArrayList;
import java.util.Collections;

public class AccountEdit extends AppCompatActivity implements View.OnClickListener, View.
OnKeyListener, FblaAzure.LogonResultListener {

    private ActivityAccountBinding mBinding;
    private ArrayAdapter<CharSequence> mStateAdapter;
    private ArrayAdapter<Schools> mSchoolAdapter;
    private ArrayList<Schools> mSchools;
    private FblaAzure mAzure;

    protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_account);
         mSchools = new ArrayList<Schools>(0);
         Bundle b = getIntent().getExtras();
         String userId = b.getString("userId");
         String token = b.getString("token");
         if (userId == null || token == null) {
             Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.LENGTH_LONG).show();
             setResult(Activity.RESULT_CANCELED, new Intent());
             finish();
             return;
         }
         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_account);
         clearSchools();
         mBinding.save.setEnabled(false);
         setSupportActionBar(mBinding.myToolbar);

         // Load the states onto the spinner from the resource file
         mStateAdapter = ArrayAdapter.createFromResource(this, R.array.states_list, android.R.layout.simple_spinner_item);
         mStateAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
         mBinding.state.setAdapter(mStateAdapter);

         // Bind the schools array to the spinner
         mSchoolAdapter = new ArrayAdapter<Schools>(this, android.R.layout.simple_spinner_item, mSchools);
         mSchoolAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
         mBinding.school.setAdapter(mSchoolAdapter);

         mBinding.zip.setOnKeyListener(this);
         mBinding.city.setOnKeyListener(this);
         mBinding.save.setOnClickListener(this);
         mBinding.cancel.setOnClickListener(this);
         mBinding.searchZip.setOnClickListener(this);
         mBinding.searchCityState.setOnClickListener(this);

         // When you select a school, the city, state, and zip fields are updated with that school's info.
         mBinding.school.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
             @Override
             public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                 Schools school = (Schools)mBinding.school.getSelectedItem();
                 mBinding.zip.setText(school.getZip());
                 mBinding.city.setText(school.getCity());
                 int spinnerPosition = mStateAdapter.getPosition(school.getStateText());
                 mBinding.state.setSelection(spinnerPosition, false);
             }
             @Override
             public void onNothingSelected(AdapterView<?> arg0) {
                 mBinding.zip.setText("");
                 mBinding.city.setText("");
                 mBinding.state.setSelection(0, false);

             }
         });

         mAzure = new FblaAzure(this);
         mAzure.setLogonListener(this);
         mAzure.doLogon(userId, token);

         int miles = mAzure.getSearchMiles(this);
         switch (miles) {
             case 5:
                 mBinding.radius5.setChecked(true);
                 break;
             case 10:
                 mBinding.radius10.setChecked(true);
                 break;
         }

         // Make sure Name is required.
         mBinding.name.addTextChangedListener(new TextWatcher() {
             public void onTextChanged(CharSequence s, int start, int before, int count) {

             }

             public void beforeTextChanged(CharSequence s, int start, int count, int after)
 {

             }

             public void afterTextChanged(Editable s) {
                 if (s.length() > 0) {
                     mBinding.save.setEnabled(true);
                 } else {
                     mBinding.save.setEnabled(false);
                 }
             }
         });
     }

     @Override
     public void onClick(View v) {
         // Clear the popup keyboard if it's there.
         View view = this.getCurrentFocus();
         if (view != null) {
             InputMethodManager imm = (InputMethodManager)getSystemService(Context.
 INPUT_METHOD_SERVICE);
             imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
         }
         // Decide which button was pressed.
         switch (v.getId()) {
             case R.id.save:
                 // The radius value is not saved in the database. It's stored on the phone.
                 int miles = mAzure.getSearchMiles(this);
                 switch (miles) {
                     case 5:
                         if (mBinding.radius10.isChecked()) {
                             mAzure.setSearchMiles(this, 10);
                         }
                         break;
                     case 10:
                         if (mBinding.radius5.isChecked()) {
                             mAzure.setSearchMiles(this, 5);
                         }
                         break;
                 }
                 // Everything else is saved to the database on the Account
                 if (mAzure.getLoggedOn()) {
                     Account account = mAzure.getAccount();
                     account.setName(mBinding.name.getText().toString());
                     Object o = mBinding.school.getSelectedItem();
                     if (o == null || ((Schools)o).getId() == "FAKE") {
                         account.setSchool(null);
                     }
                     else {
                         Schools school = (Schools)o;
                         if (account.getSchoolId() == null || !account.getSchoolId().equals(school.getId())) {
                             account.setSchool(school);
                         }
                     }

                     // The actual command to save to Azure must be done asynchronously
                     AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object, Object>() {
                         @Override
                         protected Void doInBackground(Object... params) {
                             try {
                                 FblaAzure azure = (FblaAzure)params[0];
                                 MobileServiceTable<Account> mAccountTable = azure.getClient().getTable(Account.class);
                                 Account account = azure.getAccount();
                                 mAccountTable.update(account);
                                 Log.d("AccountEdit:onClick", "AccountEdit Saved");
                                 runOnUiThread(new Runnable() {
                                     @Override
                                     public void run() {
                                         setResult(Activity.RESULT_OK, new Intent());
                                         finish();
                                     }
                                 });
                             } catch (Exception e) {
                                 Log.d("AccountEdit", e.toString());
                             }
                             return null;
                         }
                     }.execute(mAzure);
                 }

                 break;
             case R.id.cancel:
                 // This just closes the activity, returning you to YardSaleMain.
                 setResult(Activity.RESULT_CANCELED, new Intent());
                 this.finish();
                 break;
             case R.id.search_zip:
                 // Executes a search for schools based on just the zip code.
                 final String zipCode = mBinding.zip.getText().toString();
                 if (zipCode.length() > 0) {
                     clearSchools();
                     searchSchools(zipCode);
                 } else Log.d("AccountEdit:search", "FAIL");
                 break;
             case R.id.search_city_state:
                 // Executes a search for schools based on city and state.
                 int statePosition = mBinding.state.getSelectedItemPosition();
                 if (statePosition > 0) {
                     clearSchools();
                     String stateSearch = mStateAdapter.getItem(statePosition).toString();
                     String citySearch = mBinding.city.getText().toString();
                     Log.d("AccountEdit:search", citySearch+", "+stateSearch);
                     searchZip(citySearch, stateSearch);
                 } else Log.d("AccountEdit:search", "FAIL");
             default:
                 break;
         }
     }

     // This clears the schools spinner, but adds a fake "No School" entry
     private void clearSchools() {
         mSchools.clear();
         Schools fake = new Schools();
         fake.setId("FAKE");
         fake.setSchool("No School Selected");
         mSchools.add(fake);
     }

     // This removes the fake "No School" entry if it's there
     private void addSchool(Schools school) {
         if (mSchools.get(0).getId() == "FAKE") mSchools.remove(0);
         mSchools.add(school);
     }

     // This is the actual search using city and state. It actually has to do two queries.
     // First, we query the ZipCodes table for all of the zip codes matching the city and
 state.
     // Then we get all of the schools in each zip code. Finally, we sort that list and load
     // them into the array, which is bound to the spinner.
     private void searchZip(final String city, final String state) {
         AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object, Object>() {
             @Override
             protected Void doInBackground(Object... params) {
                 try {
                     FblaAzure azure = (FblaAzure)params[0];
                     // Find matching zip codes for the city and state
                     final MobileServiceList<ZipCodes> zipCodes =
                             azure.getClient().getTable(ZipCodes.class).where()
                                     .field("stateText").eq(state)
                                     .and().startsWith("city", city)
                                     .orderBy("city", QueryOrder.Ascending).execute().get();
                     ArrayList<String> uniqueZips = new ArrayList<>();
                     // Remove duplicate zip codes (some cities have multiple versions of the
  same name)
                     for (ZipCodes zip : zipCodes) {
                         if (!uniqueZips.contains(zip.getZip())) uniqueZips.add(zip.getZip()
 );
                     }
                     final ArrayList<Schools> allSchools = new ArrayList<>();
                     for (String z : uniqueZips) {
                         // Now get all of the schools in each zip code
                         final MobileServiceList<Schools> schools =
                                 azure.getClient().getTable(Schools.class).where()
                                         .field("zip").eq(z)
                                         .orderBy("school", QueryOrder.Ascending).execute().
 get();
                         for (Schools school : schools) allSchools.add(school);
                     }
                     // Sort the list
                     Collections.sort(allSchools);
                     runOnUiThread(new Runnable() {
                         @Override
                         public void run() {
                         // Put the schools into the array and notify the spinner that it's
 different.
                         for (Schools s : allSchools) {
                             addSchool(s);
                         }
                         mSchoolAdapter.notifyDataSetChanged();
                         }
                     });

                 } catch (Exception e) {
                     Log.d("SearchZip", e.toString());
                 }
                 return null;
             }
         }.execute(mAzure);
     }

     // This is a very straight forward fetch of all schools in a given zip code.
     private void searchSchools(final String zip) {
         AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object, Object>() {
             @Override
             protected Void doInBackground(Object... params) {
                 try {
                     FblaAzure azure = (FblaAzure)params[0];
                     final MobileServiceList<Schools> schools =
                             azure.getClient().getTable(Schools.class).where()
                                     .field("zip").eq(zip)
                                     .orderBy("school", QueryOrder.Ascending).execute().get(
 );
                     runOnUiThread(new Runnable() {
                         @Override
                         public void run() {
                             for (Schools s : schools) {
                                 addSchool(s);
                                 Log.d("SearchSchool", s.toString());
                             }
                             mSchoolAdapter.notifyDataSetChanged();
                         }
                     });

                 } catch (Exception e) {
                     Log.d("SearchSchool", e.toString());
                 }

                 return null;
             }
         }.execute(mAzure);
     }

     @Override
     public void onBackPressed() {
         setResult(Activity.RESULT_CANCELED, new Intent());
         this.finish();
     }

     // This allows the user to press enter on the popup keyboard and
     // have it automatically execute the corresponding search (zip or city/state).
     @Override
     public boolean onKey(View v, int keyCode, KeyEvent event) {
         if (event.getAction() == KeyEvent.ACTION_DOWN)
         {
             switch (keyCode)
             {
                 case KeyEvent.KEYCODE_DPAD_CENTER:
                 case KeyEvent.KEYCODE_ENTER:
                     switch (v.getId()) {
                         case R.id.zip:
                             this.onClick(mBinding.searchZip);
                             break;
                         case R.id.city:
                             this.onClick(mBinding.searchCityState);
                             break;
                     }
                     return true;
                 default:
                     break;
             }
         }
         return false;
     }

     @Override
     public void onLogonComplete(Exception e) {
         // Display your current school.
         Account account = mAzure.getAccount();
         mBinding.name.setText(account.getName());
         mBinding.save.setEnabled(account.getName().length() > 0);
         if (account.getSchool() != null) {
             Schools school = account.getSchool();
             addSchool(school);
             mSchoolAdapter.notifyDataSetChanged();
             mBinding.zip.setText(school.getZip());
             mBinding.city.setText(school.getCity());
             int spinnerPosition = mStateAdapter.getPosition(school.getStateText());
             mBinding.state.setSelection(spinnerPosition, false);        }
     }
 }
```

```java
/* FblaPicture.java
   =============================================================================
                            Josh Talley and Daniel O'Donnell
                                   Dulaney High School
                         Mobile Application Development 2016-17
   =============================================================================
   Purpose: This class contains a bunch of helper methods that make it easy to
   manage pictures.

   It will automatically resize large pictures so they assume the scale of
   your phone. This saves space when storing those pictures in the database.

   It also has functions that convert Bitmaps to and from an encoded string.
   The encoded string is saved in an Azure database table.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Point;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.util.Base64;
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.LinearLayout;

import com.microsoft.azure.storage.CloudStorageAccount;
import com.microsoft.azure.storage.StorageException;
import com.microsoft.azure.storage.blob.CloudBlobClient;
import com.microsoft.azure.storage.blob.CloudBlobContainer;
import com.microsoft.azure.storage.blob.CloudBlockBlob;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.URISyntaxException;
import java.security.InvalidKeyException;

public class FblaPicture {
    final private static String mStorageConnection = "DefaultEndpointsProtocol=https;
AccountName=fbla;AccountKey=
TjlylN1KDieodg23eAAgq0bV6rvLpxUM3PAAGGkjWp5Jf8XshGhr87agsbWMrYyEwgMTQ4MhoeK7L4kxdv9Agg==;
EndpointSuffix=core.windows.net";
    private static LinearLayout mLayoutImage;
    private static CloudBlobContainer mContainer = null;
    final private static Object containerLock = new Object();

    public static void setLayoutImage(LinearLayout layout)
    {
        mLayoutImage = layout;
    }
    public static int getImageHeight()
    {
        if (mLayoutImage.getHeight() > 0)
            return mLayoutImage.getHeight() / 2;
        else return 0;
    }

    // Returns dimensions of phone in pixels
    public static Point GetSize(Context c)
    {
        WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();
        Point size = new Point();
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.
HONEYCOMB_MR2)
        {
            display.getSize(size);
        }
        else // Old Version
        {
            size.set(display.getWidth(), display.getHeight());
        }
        return size;
    }

    // Loads a bitmap picture onto the ImageView item on the layout.
    public static void LoadPictureOnView(ImageView view, Bitmap original) {
        int vh = getImageHeight();
        view.setMinimumHeight(vh);
        view.setMaxHeight(vh);
        view.setImageBitmap(original);
    }

    public static Bitmap GetPictureFromView(ImageView view) {
        Drawable d = view.getDrawable();
        if (d == null) return null;
        return ((BitmapDrawable)d).getBitmap();
    }

    // Resizes a picture selected from the gallery or taken by the camera so they are a
common size.
    public static Bitmap ResizePicture(Context c, Bitmap original) {
        int w = original.getWidth();
        int h = original.getHeight();
        Point screen = GetSize(c);
        // Force everything to be 500 pixels long
        int screenL = 500;
        int originL = (w > h) ? w : h;
        int originS = (w > h) ? h : w;

        int newS = (int)((float)screenL * ((float)originS / (float)originL));
        if (w > h)
        {
            Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
 + w + "x" + h + " to " + screenL + "x" + newS);
            return Bitmap.createScaledBitmap(original, screenL, newS, true);
        }
        else
        {
            Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
 + w + "x" + h + " to " + newS + "x" + screenL);
            return Bitmap.createScaledBitmap(original, newS, screenL, true);
        }
    }

    // Uploads a picture to Azure storage. This must be run in the background.
    public static void UploadImage(String itemId, Bitmap image) throws URISyntaxException,
InvalidKeyException, StorageException, IOException {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image.compress(Bitmap.CompressFormat.PNG, 100, baos);
        byte[] b = baos.toByteArray();
        ByteArrayInputStream bs = new ByteArrayInputStream(b);
        synchronized (containerLock) {
            if (mContainer == null) {
                CloudStorageAccount storageAccount = CloudStorageAccount.parse(mStorageConnection);
                CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
                mContainer = blobClient.getContainerReference("yardsale");
            }
            CloudBlockBlob imageBlob = mContainer.getBlockBlobReference(itemId);
            imageBlob.upload(bs, b.length);
            Log.d("UploadImage", "Uploaded " + itemId);
        }
    }

    // Downloads a picture from Azure storage. This must be run in the background.
    // If the image does not exist, it will return a null.
    public static Bitmap DownloadImage(String itemId) {
        try {
            ByteArrayOutputStream bs = new ByteArrayOutputStream();
            synchronized (containerLock) {
                if (mContainer == null) {
                    CloudStorageAccount storageAccount = CloudStorageAccount.parse(mStorageConnection);
                    CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
                    mContainer = blobClient.getContainerReference("yardsale");
                }
                CloudBlockBlob blob = mContainer.getBlockBlobReference(itemId);
                if (blob.exists()) {
                    blob.download(bs);
                } else {
                    return null;
                }
            }
            // Convert to bitmap
            byte[] b = bs.toByteArray();
            return BitmapFactory.decodeByteArray(b, 0, b.length);
        } catch (Exception ex) {
            Log.d("FblaPicture:Download", itemId + " - " + ex.toString());
        }
        return null;
    }

    public static void DeleteImage(String itemId) {
        try {
            synchronized (containerLock) {
                if (mContainer == null) {
                    CloudStorageAccount storageAccount = CloudStorageAccount.parse(mStorageConnection);
                    CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
                    mContainer = blobClient.getContainerReference("yardsale");
                }
                CloudBlockBlob blob = mContainer.getBlockBlobReference(itemId);
                if (blob.exists()) {
                    blob.delete();
                }
            }
        } catch (Exception ex) {
            Log.d("FblaPicture:Delete", itemId + " - " + ex.toString());
        }
    }
}

```

```java
/* HelpAdapter.java
   =============================================================================
                        Josh Talley and Daniel O'Donnell
                               Dulaney High School
                     Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the recycler view adapter for the Help activity.
*/
package com.fbla.dulaney.fblayardsale;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class HelpAdapter  extends RecyclerView.Adapter<HelpAdapter.ViewHolder> {

    @Override
    public int getItemViewType(int position) {
        return position;
    }

    @Override
    public HelpAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v;
        switch (viewType)
        {
            case 0:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help01_register, parent, false);
                break;
            case 1:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help02_login, parent, false);
                break;
            case 2:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help03_switch, parent, false);
                break;
            case 3:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help04_editaccount, parent, false);
                break;
            case 4:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help05_addsales, parent, false);
                break;
            case 5:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help06_viewsales, parent, false);
                break;
            case 6:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help07_deletesales, parent, false);
                break;
            case 7:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help08_comment, parent, false);
                break;
            case 8:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help09_deletecomment, parent, false);
                break;
            case 9:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help10_numbersales, parent, false);
                break;
            case 10:
                v = LayoutInflater.from(parent.getContext())
                        .inflate(R.layout.help11_logout, parent, false);
                break;
            default:
                v = null;
                break;
        }
        if (v == null) return null;
        return new HelpAdapter.ViewHolder(v);
    }

    @Override
    public void onBindViewHolder(HelpAdapter.ViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 11;
    }

    class ViewHolder extends RecyclerView.ViewHolder {

        public ViewHolder(View itemView) {
            super(itemView);
        }
    }
}

```

```java
/* MapFragment.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This fragment uses the Google Map API to display and interactive
   map. The data in LocalController is used to place pins on the map of your
   school and all schools within a 10 or 5 mile radius that have at least
   one item for sale. The pins are color coded so that schools with only
   1-2 items are yellow, 3-4 items are orange, and 5+ items are red. Your school
   pin is always azure.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Context;
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.fbla.dulaney.fblayardsale.controller.LocalController;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.fbla.dulaney.fblayardsale.model.Schools;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapsInitializer;
import com.google.android.gms.maps.OnMapReadyCallback;

import android.support.v4.app.Fragment;

import com.fbla.dulaney.fblayardsale.databinding.FragmentMapBinding;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import java.util.ArrayList;
import java.util.HashMap;

public class MapFragment extends Fragment implements LocalController.RefreshResultListener {

    private MapFragment.OnFragmentInteractionListener mListener;
    FragmentMapBinding mBinding;
    private GoogleMap mMap = null;

    @Override
    public void onRefreshComplete() {
        loadMap();
    }

    public interface OnFragmentInteractionListener {
        public void onMapAttach(MapFragment f);
        public void onMapDetach(MapFragment f);
    }

    // Implementation of Fragment
    public static MapFragment newInstance(String param1, String param2) {
        MapFragment fragment = new MapFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        return fragment;
    }

    public MapFragment() {
        // Required empty public constructor
    }

    public void setEnabled(boolean enable) {

    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        try {
            mListener = (MapFragment.OnFragmentInteractionListener) context;
            mListener.onMapAttach(this);
            LocalController.attachRefreshListener(this);
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener.onMapDetach(this);
        mListener = null;
        LocalController.detachRefreshListener(this);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        mBinding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_map, container, false);
        View view = mBinding.getRoot();

        Log.d("Map:onCreateView", "Start");
        mBinding.map.onCreate(savedInstanceState);
        mBinding.map.onResume();

        try {
            MapsInitializer.initialize(getActivity().getApplicationContext());
        } catch (Exception e) {
            Log.d("Map:onCreateView", e.getMessage());
        }

        mBinding.map.getMapAsync(new OnMapReadyCallback() {
            @Override
            public void onMapReady(GoogleMap googleMap) {
                mMap = googleMap;
                Log.d("Map", "Ready");
                loadMap();
            }
        });

        return view;
    }

    private void loadMap() {
        if (mMap == null) return;
        mMap.clear();
        YardSaleMain main = (YardSaleMain)getActivity();
        FblaAzure azure = main.getAzure();
        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.getUiSettings().setAllGesturesEnabled(true);

        if (azure.getAccount() == null) return;
        Schools mySchool = azure.getAccount().getSchool();
        if (mySchool == null) return;
        // Get all of the distinct schools from the LocalController
        ArrayList<Schools> schools = new ArrayList<>();
        HashMap<String, Integer> counts = new HashMap<String, Integer>();
        for (int i = 0; i < LocalController.getItemCount(); i++) {
            SaleItem item = LocalController.getItem(i);
            Schools school = item.getAccount().getSchool();
            if (!schools.contains(school)) {
                schools.add(school);
                counts.put(school.getId(), new Integer(1));
            } else {
                counts.put(school.getId(), counts.get(school.getId()) + 1);
            }
        }
        LatLng myLL = null;
        // Mark all nearby schools on the map.
        for (Schools s : schools) {
            String title = s.getSchool();
            Integer sales = counts.get(s.getId());
            String desc = "Items for sale: " + sales.toString();
            LatLng ll = new LatLng(s.getLat(), s.getLong());
            BitmapDescriptor bm;
            if (s.getId().equals(mySchool.getId())) {
                myLL = ll;
                bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_AZURE);
            } else {
                // Change the hue of the marker depending on how many items are for sale at
 the school.
                switch (sales) {
                    case 1:
                    case 2:
                        bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_YELLOW);
                        break;
                    case 3:
                    case 4:
                        bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_ORANGE);
                        break;
                    default:
                        bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_RED);
                        break;
                }
            }
            mMap.addMarker(new MarkerOptions().position(ll)
                    .title(title).snippet(desc).icon(bm));
        }
        // See if your school has already been marked. If not, mark it.
        if (myLL == null) {
            myLL = new LatLng(mySchool.getLat(), mySchool.getLong());
            String title = mySchool.getSchool();
            String desc = "Items for sale: 0";
            mMap.addMarker(new MarkerOptions().position(myLL)
                    .title(title).snippet(desc)
                    .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_AZURE)));
        }
        // Set the camera on your school
        mMap.moveCamera(CameraUpdateFactory.newLatLng(myLL));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(12.0f));
    }

    @Override
    public void onResume() {
        super.onResume();
        Log.d("Map", "Resume");
        loadMap();
        mBinding.map.onResume();
    }

    @Override
    public void onPause() {
        super.onPause();
        mBinding.map.onPause();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        mBinding.map.onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        mBinding.map.onLowMemory();
    }

    // Initializes layout items
    @Override
    public void onActivityCreated(Bundle bundle) {
        super.onActivityCreated(bundle);
    }

}
```

```java
/* HomeFragment.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the first fragment loaded on YardSaleMain. It shows the
   application icon and is used like a menu. Buttons take you other activities.
   You can also swipe left to get to the Local Sales fragment.
*/
package com.fbla.dulaney.fblayardsale;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.fbla.dulaney.fblayardsale.databinding.FragmentHomeBinding;

public class HomeFragment extends Fragment implements View.OnClickListener {

    private OnFragmentInteractionListener mListener;
    private FragmentHomeBinding mBinding;

    @Override
    public void onClick(View v) {
        YardSaleMain main = (YardSaleMain)getActivity();
        FblaAzure azure = main.getAzure();
        boolean loggedOn = (azure != null && azure.getLoggedOn());
        switch (v.getId()) {

            case R.id.account:
                if (loggedOn) {
                    Intent i = new Intent(main, AccountEdit.class);
                    Bundle b = new Bundle();
                    b.putString("userId", azure.getUserId());
                    b.putString("token", azure.getToken());
                    i.putExtras(b);
                    getActivity().startActivityForResult(i, 0);
                }
                break;
            case R.id.add:
                if (loggedOn) {
                    Intent i = new Intent(main, AddSales.class);
                    Bundle b = new Bundle();
                    b.putString("userId", azure.getUserId());
                    b.putString("token", azure.getToken());
                    i.putExtras(b);
                    getActivity().startActivity(i);
                }
                break;
            case R.id.my:
                if (loggedOn) {
                    Intent i = new Intent(main, MySales.class);
                    Bundle b = new Bundle();
                    b.putString("userId", azure.getUserId());
                    b.putString("token", azure.getToken());
                    i.putExtras(b);
                    getActivity().startActivity(i);
                }
                break;
            case R.id.help:
                getActivity().startActivity(new Intent(getActivity(), Help.class));
                break;
            case R.id.logout:
                // Logout is a problem. Azure doesn't seem to be able to handle it
                // when I clear the cookies in order to force a new logon prompt.
                // If you try running the app too soon after logging off,
                // you get this strange net::ERR_EMPTY_RESPONSE error, which is coming
                // from the Azure library itself. Because of that problem, we are removing
                // the logoff feature and relabeling this button "Close App"
                //main.Logoff();
                getActivity().finish();
                break;
            default:
                break;
        }
    }

    public void setEnabled(boolean enable) {
        if (mBinding != null)
            mBinding.fragmentHome.setEnabled(enable);
    }

    public interface OnFragmentInteractionListener {
        public void onHomeAttach(HomeFragment f);
        public void onHomeDetach(HomeFragment f);
    }

    // Implementation of Fragment
    public static HomeFragment newInstance(String param1, String param2) {
        HomeFragment fragment = new HomeFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        return fragment;
    }

    public HomeFragment() {
        // Required empty public constructor
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        try {
            mListener = (OnFragmentInteractionListener) context;
            mListener.onHomeAttach(this);
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener.onHomeDetach(this);
        mListener = null;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        mBinding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_home, container, false);
        mBinding.account.setOnClickListener(this);
        mBinding.add.setOnClickListener(this);
        mBinding.my.setOnClickListener(this);
        mBinding.help.setOnClickListener(this);
        mBinding.logout.setOnClickListener(this);
        View view = mBinding.getRoot();

        return view;
    }

    // Initializes layout items
    @Override
    public void onActivityCreated(Bundle bundle) {
        super.onActivityCreated(bundle);
    }

}

```

```java
/* LocalAdapter.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the recycler view adapter for the Local fragment.
   It basically loads items from the LocalController onto the layout for display.
*/
package com.fbla.dulaney.fblayardsale;

import android.databinding.DataBindingUtil;
import android.graphics.Bitmap;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.fbla.dulaney.fblayardsale.controller.CommentListController;
import com.fbla.dulaney.fblayardsale.controller.LocalController;
import com.fbla.dulaney.fblayardsale.databinding.ListItemsBinding;
import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.fbla.dulaney.fblayardsale.model.Schools;

public class LocalAdapter extends RecyclerView.Adapter<LocalAdapter.ViewHolder> implements
View.OnClickListener {
    private View.OnClickListener mParentListener;
    private ListItemsBinding mBinding;
    private FblaAzure mAzure;

    public LocalAdapter (View.OnClickListener onClickListener, FblaAzure azure) {
        mParentListener = onClickListener;
        mAzure = azure;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ListItemsBinding mBinding = DataBindingUtil.inflate(
                LayoutInflater.from(parent.getContext()), R.layout.list_items, parent, false
);
        mBinding.sold.setOnClickListener(this);
        mBinding.layoutSold.setVisibility(View.GONE);
        mBinding.comments.setOnClickListener(this);
        View view = mBinding.getRoot();

        return new ViewHolder(view, mBinding);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        if (!mAzure.getLoggedOn()) return;
        SaleItem item = LocalController.getItem(position);
        if (item != null) {
            mBinding = holder.getBinding();
            mBinding.comments.setTag(position);
            mBinding.name.setText(item.getName());
            mBinding.price.setText(String.format("$%.2f", item.getPrice()));
            mBinding.description.setText(item.getDescription());
            mBinding.comments.setText("COMMENTS (" + item.getNumComments() + ")");
            Account account = item.getAccount();
            if (account != null) {
                mBinding.user.setText(account.getName());
                Schools school = account.getSchool();
                if (school != null) {
                    mBinding.address.setText(school.getFullAddress());
                    mBinding.chapter.setText(school.getSchool());
                }
            }
            Bitmap image = item.getPicture();
            if (image != null) {
                FblaPicture.setLayoutImage(mBinding.layoutPicture);
                FblaPicture.LoadPictureOnView(mBinding.picture, image);
            }
        }
    }

    @Override
    public int getItemCount() {
        return LocalController.getItemCount();
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.comments:
                if (mAzure.getLoggedOn()) {
                    int position = (int)v.getTag();
                    CommentListController.setItem(LocalController.getItem(position));
                    CommentListController.Refresh(mAzure);
                    mParentListener.onClick(v);
                }
                break;
        }
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private ListItemsBinding mBinding;

        public ViewHolder(View itemView, ListItemsBinding binding) {
            super(itemView);
            mBinding = binding;
        }

        public ListItemsBinding getBinding() {
            return mBinding;
        }
    }
}

```

```java
/* YardSaleMain.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the main startup activity. It uses the FblaPagerAdapter to manage 3 different
   activity fragments. This activity has the title bar and navigation buttons.
   The ViewPager fills the center, which holds each page fragment. It automatically handles
   swipes and smooth transitions between each page. Most navigation is handled by this activity.
   This activity will also execute the initial login using a Microsoft account.
*/

package com.fbla.dulaney.fblayardsale;

import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.util.Log;
import android.view.View;
import android.widget.Toast;

import com.fbla.dulaney.fblayardsale.controller.LocalController;
import com.fbla.dulaney.fblayardsale.controller.MySalesController;
import com.fbla.dulaney.fblayardsale.databinding.ActivityYardsaleBinding;
import com.fbla.dulaney.fblayardsale.model.Account;

public class YardSaleMain extends AppCompatActivity implements View.OnClickListener,
        HomeFragment.OnFragmentInteractionListener,
        LocalFragment.OnFragmentInteractionListener,
        MapFragment.OnFragmentInteractionListener,
        FblaAzure.LogonResultListener{

    // Class Variables
    private FblaPagerAdapter mPagerAdapter;
    public ActivityYardsaleBinding mBinding;
    private FblaAzure mAzure;
    private boolean mLogonComplete = false;
    private String mTitle;

    public void Logoff() {
        mAzure.doLogoff(this);
        //this.finish();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("YardSaleMain", "onCreate");
        setContentView(R.layout.activity_yardsale);

        mAzure = new FblaAzure(this);

        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_yardsale);
        mPagerAdapter = new FblaPagerAdapter(getSupportFragmentManager(), this);
        mBinding.pager.setAdapter(mPagerAdapter);
        mBinding.pager.addOnPageChangeListener(mPagerAdapter);
        mPagerAdapter.onPageSelected(0);
        mBinding.home.setOnClickListener(this);
        mBinding.local.setOnClickListener(this);
        mBinding.map.setOnClickListener(this);
        setSupportActionBar(mBinding.myToolbar);
        mTitle = mBinding.myToolbar.getTitle().toString();

        // Make sure everything is disabled until the logon completes
        mBinding.local.setEnabled(false);
        mBinding.map.setEnabled(false);
        mBinding.pager.setEnabled(false);

        mAzure.setLogonListener(this);
        mAzure.doLogon();
    }

    //@Override
    public void onClick(View v) {
        // Perform page changes so they transition just like a swipe.
        int pg;
        switch (v.getId()) {
            case R.id.home:
                pg = 0;
                break;
            case R.id.local:
                pg = 1;
                break;
            default:
                pg = 2;
                break;
        }
        mBinding.pager.setCurrentItem(pg, true);
    }

    public FblaAzure getAzure() { return mAzure; }

    private HomeFragment mHomeFragment = null;
    public void onHomeAttach(HomeFragment f) {
        mHomeFragment = f;
        mHomeFragment.setEnabled(mLogonComplete);
    }
    public void onHomeDetach(HomeFragment f) {
        mHomeFragment = null;
    }

    private LocalFragment mLocalFragment = null;
    public void onLocalAttach(LocalFragment f) {
        mLocalFragment = f;
        mLocalFragment.setEnabled(mLogonComplete);
    }
    public void onLocalDetach(LocalFragment f) {
        mLocalFragment = null;
    }

    private MapFragment mMapFragment = null;
    public void onMapAttach(MapFragment f) {
        mMapFragment = f;
        mMapFragment.setEnabled(mLogonComplete);
    }
    public void onMapDetach(MapFragment f) {
        mMapFragment = null;
    }

    public void onLogonComplete(Exception e) {
        if (e != null) {
            Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
            finish();
        } else if (!mLogonComplete) {
            Log.d("YardSaleMain", "Logon Complete");
            Account account = mAzure.getAccount();
            mLogonComplete = true;
            mBinding.myToolbar.setTitle(mTitle + " - " + account.getName());
            mBinding.local.setEnabled(true);
            mBinding.map.setEnabled(true);
            mBinding.pager.setEnabled(true);
            if (mHomeFragment != null) {
                mHomeFragment.setEnabled(true);
            }
            if (mLocalFragment != null) {
                mLocalFragment.setEnabled(true);
            }
            if (mMapFragment != null) {
                mMapFragment.setEnabled(true);
            }

            if (account.getName() == null || account.getName().equals("")) {
                Intent i = new Intent(this, AccountEdit.class);
                Bundle b = new Bundle();
                b.putString("userId", mAzure.getUserId());
                b.putString("token", mAzure.getToken());
                i.putExtras(b);
                this.startActivityForResult(i, 0);
            } else {
                MySalesController.Refresh(mAzure);
                LocalController.Refresh(this, mAzure);
            }
        } else {
            Account account = mAzure.getAccount();
            MySalesController.Refresh(mAzure);
            LocalController.Refresh(this, mAzure);
            mBinding.myToolbar.setTitle(mTitle + " - " + account.getName());
        }
    }

    // Require two presses on the back button to exit the activity.
    private Boolean exit = false;
    @Override
    public void onBackPressed() {
        if (exit) {
            finish(); // finish activity
        } else {
            Toast.makeText(this, "Press Back again to Exit.",
                    Toast.LENGTH_SHORT).show();
            exit = true;
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    exit = false;
                }
            }, 3 * 1000);

        }

    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            mAzure.doLoadAccount();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        mAzure = null;
    }
}

```

```java
/* LocalFragment.java
   =============================================================================
                           Josh Talley and Daniel O'Donnell
                                   Dulaney High School
                        Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the second fragment loaded on YardSaleMain. It will display
   all sale items that are within either 5 or 10 miles of your school,
   excluding any of your own sale items.

   Those sale items are loaded from the LocalController.

   You can swipe right to get to the Home fragment.
   You can swipe left to get to the Map fragment.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.Context;
import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.fbla.dulaney.fblayardsale.controller.LocalController;
import com.fbla.dulaney.fblayardsale.databinding.FragmentLocalBinding;

public class LocalFragment extends Fragment implements View.OnClickListener {

    private LocalFragment.OnFragmentInteractionListener mListener;
    private FragmentLocalBinding mBinding;

    @Override
    public void onClick(View v) {
        YardSaleMain main = (YardSaleMain)getActivity();
        FblaAzure azure = main.getAzure();
        switch (v.getId()) {
            case R.id.comments:
                Intent i = new Intent(main, Comments.class);
                Bundle b = new Bundle();
                b.putString("userId", azure.getUserId());
                b.putString("token", azure.getToken());
                i.putExtras(b);
                getActivity().startActivity(i);
                break;
            default:
                break;
        }
    }

    public void setEnabled(boolean enable) {
        if (mBinding != null)
            mBinding.fragmentLocal.setEnabled(enable);
    }

    public interface OnFragmentInteractionListener {
        public void onLocalAttach(LocalFragment f);
        public void onLocalDetach(LocalFragment f);
    }

    // Implementation of Fragment
    public static LocalFragment newInstance(String param1, String param2) {
        LocalFragment fragment = new LocalFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        return fragment;
    }

    public LocalFragment() {
        // Required empty public constructor
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        try {
            mListener = (LocalFragment.OnFragmentInteractionListener) context;
            mListener.onLocalAttach(this);
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener.onLocalDetach(this);
        mListener = null;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        mBinding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_local, container, false);
        View view = mBinding.getRoot();
        return view;
    }

    @Override
    public void onActivityCreated(Bundle bundle) {
        super.onActivityCreated(bundle);
        // Setup the RecyclerView here because the data changes.
        YardSaleMain mParent = (YardSaleMain)getActivity();
        mBinding.list.setLayoutManager(new LinearLayoutManager(mParent));
        LocalAdapter adapter = new LocalAdapter(this, mParent.getAzure());
        LocalController.AttachAdapter(adapter);
        LocalController.Refresh(mParent, mParent.getAzure());
        mBinding.list.setAdapter(adapter);
    }

}

```

```java
/* MySalesAdapter.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This is the recycler view adapter for the MySales fragment.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.DialogInterface;
import android.databinding.DataBindingUtil;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.fbla.dulaney.fblayardsale.controller.CommentListController;
import com.fbla.dulaney.fblayardsale.controller.MySalesController;
import com.fbla.dulaney.fblayardsale.databinding.ListItemsBinding;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

public class MySalesAdapter extends RecyclerView.Adapter<MySalesAdapter.ViewHolder>
implements View.OnClickListener {
    private View.OnClickListener mParentListener;
    private ListItemsBinding mBinding;
    private MySales mContext;
    FblaAzure mAzure;

    public MySalesAdapter (MySales context, View.OnClickListener onClickListener, FblaAzure
azure) {
        mContext = context;
        mParentListener = onClickListener;
        mAzure = azure;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ListItemsBinding mBinding = DataBindingUtil.inflate(
                LayoutInflater.from(parent.getContext()), R.layout.list_items, parent, false
);
        mBinding.sold.setOnClickListener(this); // This really just deletes the item.
        mBinding.comments.setOnClickListener(this);
        mBinding.layoutAddress.setVisibility(View.GONE);
        mBinding.layoutChapter.setVisibility(View.GONE);
        mBinding.layoutUser.setVisibility(View.GONE);
        View view = mBinding.getRoot();

        return new ViewHolder(view, mBinding);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        if (!mAzure.getLoggedOn()) return;
        SaleItem item = MySalesController.getItem(position);
        if (item != null) {
            mBinding = holder.getBinding();
            mBinding.comments.setTag(position);
            mBinding.name.setText(item.getName());
            mBinding.price.setText(String.format("$%.2f", item.getPrice()));
            mBinding.description.setText(item.getDescription());
            mBinding.comments.setText("COMMENTS (" + item.getNumComments() + ")");
            mBinding.sold.setTag(position); // Being sold means to delete it.
            if (item.getHasPicture()) {
                Bitmap image = item.getPicture();
                FblaPicture.setLayoutImage(mBinding.layoutPicture);
                FblaPicture.LoadPictureOnView(mBinding.picture, image);
            }
        }
    }

    @Override
    public int getItemCount() {
        return MySalesController.getItemCount();
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.comments:
                if (mAzure.getLoggedOn()) {
                    int position = (int)v.getTag();
                    SaleItem item = MySalesController.getItem(position);
                    CommentListController.setItem(item);
                    CommentListController.Refresh(mAzure);
                    Log.d("MySalesAdapter", "Refreshed for " + position);
                    mParentListener.onClick(v);
                }
                break;
            case R.id.sold:
                // When it's sold, we just delete it.
                final int position = (int)v.getTag();
                AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
                builder.setTitle("Are You Sure?");
                final TextView info = new TextView(mContext);
                info.setText("By Pressing Confirm, The Item Will Be Deleted.");
                info.setPadding(30, 0, 0, 0);
                builder.setView(info);

                builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener()
 {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Log.d("MySalesAdapter", "delete");
                        deleteItem(position);
                        dialog.dismiss();
                    }
                });

                builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });

                builder.show();
                break;
            default:
                break;
        }
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private ListItemsBinding mBinding;

        public ViewHolder(View itemView, ListItemsBinding binding) {
            super(itemView);
            mBinding = binding;
        }

        public ListItemsBinding getBinding() {
            return mBinding;
        }
    }

    private void deleteItem(int position) {
        if (!mAzure.getLoggedOn()) return;

        final int pos = position;
        final SaleItem item = MySalesController.getItem(position);
        final MobileServiceTable<SaleItem> mSaleItemTable = mAzure.getClient().getTable(
SaleItem.class);
        // Delete the comment from the database.
        AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
            @Override
            protected Void doInBackground(Void... params) {
                try {
                    FblaPicture.DeleteImage(item.getId());
                    mSaleItemTable.delete(item);
                    Log.d("MySales:delete", "Deleted item " + item.getName());
                    mContext.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            MySalesController.removeItem(pos);
                        }
                    });
                } catch (Exception e) {
                    Log.d("MySales:delete", e.toString());
                }
                return null;
            }
        };
        task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
    }
}

```

```java
/* CommentsAdapter.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: This adapter is used by the Comments activity to manage the list of
   comments. It makes use of the CommentListController.

   When you delete a comment, it uses a popup window to ask if you are sure.
*/
package com.fbla.dulaney.fblayardsale;

import android.content.DialogInterface;
import android.databinding.DataBindingUtil;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.fbla.dulaney.fblayardsale.controller.CommentListController;
import com.fbla.dulaney.fblayardsale.databinding.ListCommentsBinding;
import com.fbla.dulaney.fblayardsale.model.ItemComment;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

public class CommentsAdapter extends RecyclerView.Adapter<CommentsAdapter.ViewHolder>
implements View.OnClickListener {
    private View.OnClickListener mParentListener;
    private ListCommentsBinding mBinding;
    private Comments mContext;
    private FblaAzure mAzure;

    public CommentsAdapter (Comments context, View.OnClickListener onClickListener,
FblaAzure azure) {
        mContext = context;
        mParentListener = onClickListener;
        mAzure = azure;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ListCommentsBinding mBinding = DataBindingUtil.inflate(
                LayoutInflater.from(parent.getContext()), R.layout.list_comments, parent,
false);
        View view = mBinding.getRoot();
        mBinding.delete.setOnClickListener(this);

        Log.d("CommentsAdapter", "onCreateViewHolder");
        return new ViewHolder(view, mBinding);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        if (!mAzure.getLoggedOn()) return;
        ItemComment comment = CommentListController.getComment(position);
        if (comment != null) {
            mBinding = holder.getBinding();
            Log.d("CommentsAdapter", "onBindViewHolder");
            mBinding.comments.setText(comment.getComment());
            if (comment.getAccount() == null) mBinding.username.setText("{Unknown}");
            else mBinding.username.setText(comment.getAccount().getName());
            mBinding.delete.setTag(position);
        }
    }

    @Override
    public int getItemCount() {
        return CommentListController.getCommentCount();
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.delete:
                final int position = (int)v.getTag();
                AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
                builder.setTitle("Are You Sure?");
                final TextView info = new TextView(mContext);
                info.setText("By Pressing Confirm, The Comment Will Be Deleted.");
                info.setPadding(30, 0, 0, 0);
                builder.setView(info);

                builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener()
{
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        deleteComment(position);
                        dialog.dismiss();
                    }
                });

                builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });

                builder.show();
                break;
            default:
                break;
        }
    }

    private void deleteComment(int position) {
        if (!mAzure.getLoggedOn()) return;

        final int pos = position;
        final ItemComment comment = CommentListController.getComment(position);
        final MobileServiceTable<ItemComment> mCommentTable = mAzure.getClient().getTable(
ItemComment.class);
        // Delete the comment from the database.
        AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
            @Override
            protected Void doInBackground(Void... params) {
                try {
                    mCommentTable.delete(comment);
                    Log.d("Comments:delete", "Deleted comment " + comment.getComment());
                    mContext.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            CommentListController.removeComment(pos);
                        }
                    });
                } catch (Exception e) {
                    Log.d("Comments:delete", e.toString());
                }
                return null;
            }
        };
        task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private ListCommentsBinding mBinding;

        public ViewHolder(View itemView, ListCommentsBinding binding) {
            super(itemView);
            mBinding = binding;
        }

        public ListCommentsBinding getBinding() {
            return mBinding;
        }
    }
}

```

```java
/* FblaPagerAdapter.java
   ============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   ============================================================================
   Purpose: This simply loads the appropriate fragment onto the YardSaleMain activity.

   It also slightly adjusts the color of the button icons that represent each
   fragment (appearing as tabs).
*/
package com.fbla.dulaney.fblayardsale;

import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.util.Log;

public class FblaPagerAdapter extends FragmentStatePagerAdapter implements ViewPager.
OnPageChangeListener {
    protected YardSaleMain mContext;

    public FblaPagerAdapter(FragmentManager fm, YardSaleMain context)
    {
        super(fm);
        mContext = context;
    }

    @Override
    public Fragment getItem(int position) {
        Fragment fragment;
        switch (position)
        {
            case 0:
                fragment = new HomeFragment();
                break;
            case 1:
                fragment = new LocalFragment();
                break;
            default:
                fragment = new MapFragment();
                break;
        }
        Bundle args = new Bundle();
        args.putInt("page_position", position);

        fragment.setArguments(args);

        return fragment;
    }

    @Override
    public int getCount() {
        return 3;
    }

    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels)
 {

    }

    @Override
    public void onPageSelected(int position) {
        switch (position) {
            case 0: // Home
                mContext.mBinding.home.setTextColor(Color.BLACK);
                mContext.mBinding.local.setTextColor(Color.DKGRAY);
                mContext.mBinding.map.setTextColor(Color.DKGRAY);
                break;
            case 1: // Local
                mContext.mBinding.home.setTextColor(Color.DKGRAY);
                mContext.mBinding.local.setTextColor(Color.BLACK);
                mContext.mBinding.map.setTextColor(Color.DKGRAY);
                break;
            case 2: // Map
                mContext.mBinding.home.setTextColor(Color.DKGRAY);
                mContext.mBinding.local.setTextColor(Color.DKGRAY);
                mContext.mBinding.map.setTextColor(Color.BLACK);
                break;
            default:
                Log.d("FblaPager:Selected", "Other");
                break;
        }
    }

    @Override
    public void onPageScrollStateChanged(int state) {

    }
}

```

```java
/* Account.java
   ==============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   ==============================================================================
   Purpose: Model of the Azure database table for user account information.
   This class is used by the Azure library to query and create data in the
   Account database table.

   The link to the Schools table is also represented by holding a copy of
   the whole Schools object.
*/
package com.fbla.dulaney.fblayardsale.model;

public class Account {
    // Database Columns
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique id for the user, as provided by the Microsoft logon
    @com.google.gson.annotations.SerializedName("name")
    private String mName; // Your username
    @com.google.gson.annotations.SerializedName("schoolid")
    private String mSchoolId; // Foreign key to the selected school.

    // Transient fields will not get queried or saved to the database
    @com.google.gson.annotations.Expose(serialize = false)
    private transient Schools mSchool;

    public Account() {
        mId = "";
        mName = "";
        mSchoolId = null;
        mSchool = null;
    }

    @Override
    public String toString() {
        return getId();
    }

    // Getters and Setters
    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getName() { return mName; }
    public final void setName(String name) { mName = name; }
    public String getSchoolId() { return mSchoolId; }

    public Schools getSchool() { return mSchool; }
    public final void setSchool(Schools school) {
        mSchool = school;
        if (school == null) mSchoolId = null;
        else mSchoolId = school.getId();
    }

    @Override
    public boolean equals(Object o) {
        return o instanceof Account && ((Account)o).mId == mId;
    }
}

```

```java
/* Schools.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: Model of the Azure database table for school information.
   This class is used by the Azure library to query and create data in the
   Schools database table.

   In order to make it easy for users to select their school, all public and
   private schools in the USA and its territories were downloaded from the
   National Center for Education Statistics and loaded into the Schools database.
   https://nces.ed.gov/ccd/pubschuniv.asp
   https://nces.ed.gov/surveys/pss/pssdata.asp

*/
package com.fbla.dulaney.fblayardsale.model;

import android.support.annotation.NonNull;

public class Schools implements Comparable<Schools> {
    // Database Columns
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique ID of the school, assigned by the National Center for
Education Statistics
    @com.google.gson.annotations.SerializedName("zip")
    private String mZip; // Zip code of the school
    @com.google.gson.annotations.SerializedName("school")
    private String mSchool; // Name of the school
    @com.google.gson.annotations.SerializedName("address")
    private String mAddress; // Address of the school
    @com.google.gson.annotations.SerializedName("city")
    private String mCity; // City of the school
    @com.google.gson.annotations.SerializedName("stateText")
    private String mStateText; // State or Territory (full name, not abbreviated)
    @com.google.gson.annotations.SerializedName("lat")
    private double mLat; // Latitude
    @com.google.gson.annotations.SerializedName("long")
    private double mLong; // Longitude

    public Schools() {
        mId = "";
        mZip = "";
        mSchool = "";
        mAddress = "";
        mCity = "";
        mStateText = "";
        mLat = 0;
        mLong = 0;
    }

    @Override
    public String toString() {
        return getSchool();
    }

    //Getters and Setters
    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getZip() { return mZip; }
    public final void setZip(String zip) { mZip = zip; }
    public String getSchool() { return mSchool; }
    public final void setSchool(String school) { mSchool = school; }
    public String getAddress() { return mAddress; }
    public final void setAddress(String address) { mAddress = address; }
    public String getCity() { return mCity; }
    public final void setCity(String city) { mCity = city; }
    public String getStateText() { return mStateText; }
    public final void setStateText(String stateText) { mStateText = stateText; }
    public double getLat() { return mLat; }
    public final void setLat(double lat) { mLat = lat; }
    public double getLong() { return mLong; }
    public final void setLong(double lng) { mLong = lng; }

    // Separate full address for displaying with an item.
    public String getFullAddress() {
        return mAddress + ", " + mCity + ", " + mStateText;
    }

    @Override
    public boolean equals(Object o) {
        return o instanceof Schools && ((Schools)o).mId == mId;
    }

    // Implements Comparable so we can sort them during a city/state search.
    @Override
    public int compareTo(@NonNull Schools o) {
        return this.getSchool().compareTo(o.getSchool());
    }
}

```

```java
/* SaleItem.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: Model of the Azure database table for sale item information.
   This class is used by the Azure library to query and create data in the
   SaleItem database table.

   We store the number of comments and a link to the Account object
   so that additional details can be displayed.
*/
package com.fbla.dulaney.fblayardsale.model;

import android.graphics.Bitmap;
import com.fbla.dulaney.fblayardsale.FblaPicture;

public class SaleItem {
    // Database Columns
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique value created as a random UUID.
    @com.google.gson.annotations.SerializedName("userid")
    private String mUserId; // Foreign key to the Account.
    @com.google.gson.annotations.SerializedName("name")
    private String mName; // Name of the item.
    @com.google.gson.annotations.SerializedName("description")
    private String mDescription; // Description of the item.
    @com.google.gson.annotations.SerializedName("price")
    private float mPrice; // Price of the item.
    @com.google.gson.annotations.SerializedName("hasPicture")
    private boolean mHasPicture; // If a picture has been added or not.

    // Transient fields will not get queried or saved to the database
    @com.google.gson.annotations.Expose(serialize = false)
    private transient Bitmap mPicture;
    @com.google.gson.annotations.Expose(serialize = false)
    private transient int mNumComments; // Number of comments
    @com.google.gson.annotations.Expose(serialize = false)
    private transient Account mAccount;

    public SaleItem() {
        mAccount = null;
        mName = "";
        mId = "";
        mUserId = null;
        mDescription = "";
        mPicture = null;
        mPrice = 0;
        mNumComments = 0;
        mHasPicture = false;
    }

    @Override
    public String toString() {
        return getId();
    }

    // Getters and Setters
    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getName() { return mName; }
    public final void setName(String name) { mName = name; }
    public String getDescription() { return mDescription; }
    public final void setDescription(String description) { mDescription = description; }
    public float getPrice() { return mPrice; }
    public final void setPrice(float price) { mPrice = price; }
    public boolean getHasPicture() { return mHasPicture; }
    public final void setHasPicture(boolean hasPicture) { mHasPicture = hasPicture; }
    public Bitmap getPicture() { return mPicture; }
    public final void setPicture(Bitmap image) {
        mPicture = image;
        mHasPicture = (image != null);
    }
    public Account getAccount() { return mAccount; }
    // Setting the Account will automatically set the database foreign key, too.
    public final void setAccount(Account account) {
        mAccount = account;
        mUserId = account.getId();
    }
    public int getNumComments() { return mNumComments; }
    public final void setNumComments(int numComments) {
        mNumComments = numComments;
    }

    @Override
    public boolean equals(Object o) {
        return o instanceof SaleItem && ((SaleItem)o).mId == mId;
    }
}

```

```java
/* ZipCodes.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: Model of the Azure database table for all US zip codes.
   This class is used by the Azure library to query and create data in the
   ZipCodes database table.

   The ZipCodes database table is used to find schools. We can easily get a list
   of each school in a zip code, because the Schools table has the zip code.
   However, not everybody remembers the zip code for their school. The ZipCodes
   table was populated with the free ZipCode database containing all locations
   from this web site: http://federalgovernmentzipcodes.us

   The user selects a state, and types in the beginning of whatever city they
   want, and we can get a list of all zip codes that match.
*/
package com.fbla.dulaney.fblayardsale.model;

public class ZipCodes {
    // Database Columns
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique id assigned by the database
    @com.google.gson.annotations.SerializedName("zip")
    private String mZip; // Zip code
    @com.google.gson.annotations.SerializedName("zipType")
    private String mZipType; // Type of zip code (PO BOX, STANDARD, UNIQUE)
    @com.google.gson.annotations.SerializedName("city")
    private String mCity; // City
    @com.google.gson.annotations.SerializedName("state")
    private String mState; // State, abbreviation
    @com.google.gson.annotations.SerializedName("locationType")
    private String mLocationType; // Location Type (PRIMARY, ACCEPTABLE, NOT ACCEPTABLE)
    @com.google.gson.annotations.SerializedName("locationText")
    private String mLocationText; // Camel Case version of city and state
    @com.google.gson.annotations.SerializedName("lat")
    private double mLat; // Latitude
    @com.google.gson.annotations.SerializedName("long")
    private double mLong; // Longitude
    @com.google.gson.annotations.SerializedName("stateText")
    private String mStateText; // State, fully spelled out

    public ZipCodes() {
        mId = "";
        mZip = "";
        mZipType = "";
        mCity = "";
        mState = "";
        mLocationType = "";
        mLocationText = "";
        mLat = 0;
        mLong = 0;
        mStateText = "";
    }

    @Override
    public String toString() {
        return getZip();
    }

    //Getters and Setters
    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getZip() { return mZip; }
    public final void setZip(String zip) { mZip = zip; }
    public String getZipType() { return mZipType; }
    public final void setZipType(String zipType) { mZipType = zipType; }
    public String getCity() { return mCity; }
    public final void setCity(String city) { mCity = city; }
    public String getState() { return mState; }
    public final void setState(String state) { mState = state; }
    public String getLocationType() { return mLocationType; }
    public final void setLocationType(String locationType) { mLocationType = locationType;
}
    public String getLocationText() { return mLocationText; }
    public final void setLocationText(String locationText) { mLocationText = locationText;
}
    public double getLat() { return mLat; }
    public final void setLat(double lat) { mLat = lat; }
    public double getLong() { return mLong; }
    public final void setLong(double lng) { mLong = lng; }
    public String getStateText() { return mStateText; }
    public final void setStateText(String stateText) { mStateText = stateText; }

    @Override
    public boolean equals(Object o) {
        return o instanceof ZipCodes && ((ZipCodes)o).mId == mId;
    }
}

```

```java
/* ItemComment.java
   ============================================================================
                          Josh Talley and Daniel O'Donnell
                                 Dulaney High School
                       Mobile Application Development 2016-17
   ============================================================================
   Purpose: Model of the Azure database table for item comment information.
   This class is used by the Azure library to query and create data in the
   ItemComment database table.

   The link to the Account table is also represented by holding a copy of
   the whole Account object.
*/
package com.fbla.dulaney.fblayardsale.model;

public class ItemComment {
    // Database Columns
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique id assigned by the database.
    @com.google.gson.annotations.SerializedName("userid")
    private String mUserId; // Foreign key to the Account
    @com.google.gson.annotations.SerializedName("itemid")
    private String mItemId; // Foreign key to the SaleItem
    @com.google.gson.annotations.SerializedName("comment")
    private String mComment; // This is the actual comment text

    // Transient fields will not get queried or saved to the database
    @com.google.gson.annotations.Expose(serialize = false)
    private transient Account mAccount; // Needed to display the username

    public ItemComment() {
        mAccount = null;
        mId = "";
        mUserId = "";
        mItemId = "";
        mComment = "";
    }

    @Override
    public String toString() {
        return getId();
    }

    // Getters and Setters
    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getUserId() { return mUserId; }
    public final void setUserId(String userId) { mUserId = userId; }
    public String getItemId() { return mItemId; }
    public final void setItemId(String itemId) { mItemId = itemId; }
    public String getComment() { return mComment; }
    public final void setComment(String comment) { mComment = comment; }
    public Account getAccount() { return mAccount; }
    public final void setAccount(Account account) { mAccount = account; }

    @Override
    public boolean equals(Object o) {
        return o instanceof ItemComment && ((ItemComment)o).mId == mId;
    }
}

```

```java
/* SchoolDistance.java
   ==============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   ==============================================================================
   Purpose: Model of the Azure database that holds distances between schools.
   This class is used by the Azure library to query and create data in the
   SchoolDistance database table.

   The SchoolDistance table has been preloaded with the distance between every
   school within 10 miles of each other. The following SQL was used to do this,
   executed for each state/territory. When we tried to run everything at once,
   we decided to cancel after 5 hours and run it in chucks, per state. It took
   anywhere from 30 minutes to 6 minutes for each state/territory.

   The distance is calculated using the Haversine formula, with the SQL itself
   developed by Dayne Batten.
   http://daynebatten.com/2015/09/latitude-longitude-distance-sql/

   INSERT INTO SchoolDistance (fromid, toid, miles)
   SELECT f.id, t.id
        , 2 * 3961 * asin(sqrt(
            square(sin(radians((t.lat - f.lat) / 2))) +
            cos(radians(f.lat)) * cos(radians(t.lat)) *
            square(sin(radians((t.long - f.long) / 2)))
          )) miles
   FROM Schools f
   INNER JOIN Schools t ON (t.id <> f.id AND t.lat <> 0 AND t.long <> 0)
   WHERE 2 * 3961 * asin(sqrt(
            square(sin(radians((t.lat - f.lat) / 2))) +
            cos(radians(f.lat)) * cos(radians(t.lat)) *
            square(sin(radians((t.long - f.long) / 2)))
          )) <= 10
   AND f.stateText = @state;

*/
package com.fbla.dulaney.fblayardsale.model;

public class SchoolDistance {
    @com.google.gson.annotations.SerializedName("id")
    private String mId; // Unique id assigned by the database. We don't use this.
    @com.google.gson.annotations.SerializedName("fromid")
    private String mFromId; // Foreign key to the user's school.
    @com.google.gson.annotations.SerializedName("toid")
    private String mToId; // Foreign key to the school that's nearby.
    @com.google.gson.annotations.SerializedName("miles")
    private float mMilesId; // Distance in miles between the schools.

    @Override
    public String toString() {
        return mId;
    }

    public String getId() { return mId; }
    public final void setId(String id) { mId = id; }
    public String getFromId() { return mFromId; }
    public final void setFromId(String id) { mFromId = id; }
    public String getToId() { return mToId; }
    public final void setToId(String id) { mToId = id; }
    public float getMiles() { return mMilesId; }
    public final void setMiles(float miles) { mMilesId = miles; }

    @Override
    public boolean equals(Object o) {
        return o instanceof SchoolDistance && ((SchoolDistance)o).mId == mId;
    }
}

```

```java
/* LocalController.java
   ============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   ============================================================================
   Purpose: Used by LocalFragment to control access to the list of Sale Items in
   the user's local area (by distance from their school). Attaching a recycler
   view to the class so that when the list of items is refreshed or changed, the
   recycler view is notified of that change.

   Getting the list of nearby schools is very complicated. The Schools database
   table has been loaded with all public and private schools in the USA and its
   territories, including each school's latitude and longitude. Another table,
   called SchoolDistance, has the distance of every school within a 10-mile
   circle. This has been pre-calculated so that the query is very fast, and is
   why you are limited to either a 5-mile radius or 10-mile radius.

   We start with the school selected by the user from the Accounts page. All
   nearby schools are fetched from the SchoolDistance table. Details for each
   school is fetched from the Schools table, because we need to display those
   details with each item. Then we have to fetch all users currently tied to
   each school from the Account table. Then we fetch all items for each user
   from the SaleItem table (excluding your own). Finally, we count the number
   of comments on each item using the ItemComment table, so that it's
   displayed on the comments button.

   SchoolDistance -> Schools -> Account -> SaleItem -> ItemComment

   Finally, for each SaleItem, we download a picture (if it has one) from Azure
   storage, using FblaPicture.
*/
package com.fbla.dulaney.fblayardsale.controller;

import android.content.Context;
import android.os.AsyncTask;
import android.support.v7.widget.RecyclerView;
import android.util.Log;

import com.fbla.dulaney.fblayardsale.FblaAzure;
import com.fbla.dulaney.fblayardsale.FblaPicture;
import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.ItemComment;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.fbla.dulaney.fblayardsale.model.SchoolDistance;
import com.fbla.dulaney.fblayardsale.model.Schools;
import com.microsoft.windowsazure.mobileservices.MobileServiceList;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

import java.util.ArrayList;

public class LocalController {
    private static ArrayList<SaleItem> mSaleItems = new ArrayList<>();
    private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();

    public static void AttachAdapter(RecyclerView.Adapter adapter) {
        mAdapters.add(adapter);
    }

    public static int getItemCount() {
        return mSaleItems.size();
    }

    public static SaleItem getItem(int position) {
        if (mSaleItems.size() > position) return mSaleItems.get(position);
        else return null;
    }

    public static void notifyItem(SaleItem item) {
        if (mSaleItems.contains(item)) {
            int position = mSaleItems.indexOf(item);
            for (RecyclerView.Adapter adapter : mAdapters) {
                adapter.notifyItemChanged(position);
            }
        }
    }

    public static void addItem(SaleItem item) {
        mSaleItems.add(item);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
    }

    public static void removeItem(int position) {
        mSaleItems.remove(position);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
    }

    /*
       The Refresh executes a new search. It can be called from anywhere.
       For example, when you change your school, we have to refresh.
    */
    private static MobileServiceTable<SchoolDistance> mSchoolDistanceTable;
    private static MobileServiceTable<Schools> mSchoolsTable;
    private static MobileServiceTable<Account> mAccountTable;
    private static MobileServiceTable<SaleItem> mSaleItemTable;
    private static MobileServiceTable<ItemComment> mItemCommentTable;
    public static void Refresh(Context context, FblaAzure azure) {
        if (!azure.getLoggedOn()) return;
        mSaleItems.clear();

        final int searchMiles = azure.getSearchMiles(context);
        Account myAccount = azure.getAccount();
        if (myAccount.getSchool() == null) {
            for (RecyclerView.Adapter adapter : mAdapters) {
                adapter.notifyDataSetChanged();
            }
            return;
        }
        final String searchUserId = azure.getUserId();
        final Schools searchSchool = myAccount.getSchool();
        Log.d("LocalController:Refresh", searchSchool.getId()+" "+searchMiles);

        mSchoolDistanceTable = azure.getClient().getTable(SchoolDistance.class);
        mSchoolsTable = azure.getClient().getTable(Schools.class);
        mAccountTable = azure.getClient().getTable(Account.class);
        mSaleItemTable = azure.getClient().getTable(SaleItem.class);
        mItemCommentTable = azure.getClient().getTable(ItemComment.class);
        new AsyncTask<Object, Object, Object>() {
            @Override
            protected Object doInBackground(Object... params) {
                try {
                    ArrayList<SaleItem> saleItems = new ArrayList<>();
```

```java
                    // First get all of the schools nearby
                    Log.d("LocalController:Refresh", "Starting");
                    final MobileServiceList<SchoolDistance> distances =
                            mSchoolDistanceTable.where().field("fromid").eq(searchSchool.getId())
                                    .and().field("miles").le(searchMiles)
                                    .select("id", "fromid", "toid", "miles")
                                    .execute().get();
                    for (SchoolDistance toSchool : distances) {
                        // Get each school details
                        final Schools school = mSchoolsTable.lookUp(toSchool.getToId()).get();
                        // Get all accounts for each school
                        final MobileServiceList<Account> accounts =
                                mAccountTable.where().field("schoolid").eq(school.getId()).execute().get();
                        for (Account account : accounts) {
                            // Now get all the items for each account (excluding your own)
                            if (!account.getId().equals(searchUserId)) {
                                account.setSchool(school);
                                final MobileServiceList<SaleItem> items =
                                        mSaleItemTable.where().field("userid").eq(account.getId()).execute().get();
                                for (SaleItem item : items) {
                                    item.setAccount(account);
                                    // Get its picture
                                    if (item.getHasPicture())
                                        item.setPicture(FblaPicture.DownloadImage(item.getId()));
                                    // Finally, count the number of comments that are on each item.
                                    final MobileServiceList<ItemComment> cnt =
                                            mItemCommentTable.where().field("itemid").eq(item.getId()).includeInlineCount().execute().get();
                                    item.setNumComments(cnt.getTotalCount());
                                    saleItems.add(item);
                                }
                            }
                        }
                    }
                    return saleItems;
                } catch (Exception exception) {
                    Log.e("LocalController:Refresh", exception.toString());
                }
                return null;
            }
            @Override
            protected void onPostExecute(Object result) {
                Log.d("LocalController:Refresh", "Complete");
                if (result != null) {
                    ArrayList<SaleItem> saleItems = (ArrayList<SaleItem>)result;
                    for (SaleItem item : saleItems) {
                        mSaleItems.add(item);
                    }
                    for (RecyclerView.Adapter adapter : mAdapters) {
                        adapter.notifyDataSetChanged();
                    }
                    for (RefreshResultListener l : mListeners) {
                        l.onRefreshComplete();
                    }
                }
            }
        }.execute();
```

```java
    }

    private static ArrayList<RefreshResultListener> mListeners = new ArrayList<>();
    // Add a listener to call after refresh is complete
    public static void attachRefreshListener(RefreshResultListener listener) {
        mListeners.add(listener);
    }
    public static void detachRefreshListener(RefreshResultListener listener) {
        mListeners.remove(listener);
    }

    // This is the interface to use on the logon callbacks.
    public interface RefreshResultListener {
        void onRefreshComplete();
    }
}

```

```java
/* MySalesController.java
   ==============================================================================
                             Josh Talley and Daniel O'Donnell
                                     Dulaney High School
                          Mobile Application Development 2016-17
   ==============================================================================
   Purpose: Used by MySalesFragment to control access to the list of Sale Items owned
   by the user. Attaching a recycler view to the class so that when the list of
   items is refreshed or changed, the recycler view is notified of that change.

   Items are fetched from the SaleItem table. Then for each item, the number of
   comments are counted from the ItemComment table in order to display it on
   the comments button.

   Pictures for each item are fetched from Azure storage using FblaPicture.

*/
package com.fbla.dulaney.fblayardsale.controller;

import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.support.v7.widget.RecyclerView;
import android.util.Log;

import com.fbla.dulaney.fblayardsale.FblaAzure;
import com.fbla.dulaney.fblayardsale.FblaPicture;
import com.fbla.dulaney.fblayardsale.model.ItemComment;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.microsoft.windowsazure.mobileservices.MobileServiceList;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

import java.util.ArrayList;

public class MySalesController {
    private static ArrayList<SaleItem> mSaleItems = new ArrayList<>();
    private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();

    public static void AttachAdapter(RecyclerView.Adapter adapter) {
        mAdapters.add(adapter);
    }

    public static int getItemCount() {
        return mSaleItems.size();
    }

    public static SaleItem getItem(int position) {
        if (mSaleItems.size() > position) return mSaleItems.get(position);
        else return null;
    }

    public static void notifyItem(SaleItem item) {
        if (mSaleItems.contains(item)) {
            int position = mSaleItems.indexOf(item);
            for (RecyclerView.Adapter adapter : mAdapters) {
                adapter.notifyItemChanged(position);
            }
        }
    }

    public static void addItem(SaleItem item) {
        mSaleItems.add(item);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
    }

    public static void removeItem(int position) {
        mSaleItems.remove(position);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
    }

    private static MobileServiceTable<SaleItem> mSaleItemTable;
    private static MobileServiceTable<ItemComment> mItemCommentTable;
    public static void Refresh(FblaAzure azure) {
        Log.d("MySalesController", "Refresh");
        if (!azure.getLoggedOn()) return;
        mSaleItems.clear();

        mSaleItemTable = azure.getClient().getTable(SaleItem.class);
        mItemCommentTable = azure.getClient().getTable(ItemComment.class);
        new AsyncTask<Object, Object, Object>() {
            class TaskResult {
                public FblaAzure azure;
                public ArrayList<SaleItem> saleItems;

                public TaskResult (FblaAzure a) {
                    azure = a;
                    saleItems = new ArrayList<>();
                }
            }

            @Override
            protected Object doInBackground(Object... params) {
                try {
                    FblaAzure azure = (FblaAzure)params[0];
                    TaskResult taskResult = new TaskResult(azure);
                    final MobileServiceList<SaleItem> result =
                            mSaleItemTable.where().field("userid").eq(azure.getUserId()).
execute().get();
                    for (SaleItem s : result) {
                        final MobileServiceList<ItemComment> cnt =
                                mItemCommentTable.where().field("itemid").eq(s.getId()).
includeInlineCount().execute().get();
                        s.setNumComments(cnt.getTotalCount());
                        // Now get the picture, if it exists.
                        if (s.getHasPicture())
                            s.setPicture(FblaPicture.DownloadImage(s.getId()));
                        taskResult.saleItems.add(s);
                    }
                    return taskResult;
                } catch (Exception exception) {
                    Log.e("MySalesController", exception.toString());
                    return null;
                }
            }
            @Override
            protected void onPostExecute(Object result) {
                if (result != null) {
                    TaskResult taskResult = (TaskResult)result;
                    for (SaleItem item : taskResult.saleItems) {
                        item.setAccount(taskResult.azure.getAccount());
                        mSaleItems.add(item);
                    }
                    Log.d("MySalesController", "Set Notify");
                    for (RecyclerView.Adapter adapter : mAdapters) {
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        }.execute(azure);
    }
}

```

```java
/* CommentListController.java
   =============================================================================
                         Josh Talley and Daniel O'Donnell
                                Dulaney High School
                      Mobile Application Development 2016-17
   =============================================================================
   Purpose: Used by CommentList to control access to the list of comments for
   a selected item. Attaching a recycler view to the class so that when the list of
   items is refreshed or changed, the recycler view is notified of that change.
*/
package com.fbla.dulaney.fblayardsale.controller;

import android.os.AsyncTask;
import android.support.v7.widget.RecyclerView;
import android.util.Log;

import com.fbla.dulaney.fblayardsale.FblaAzure;
import com.fbla.dulaney.fblayardsale.model.Account;
import com.fbla.dulaney.fblayardsale.model.ItemComment;
import com.fbla.dulaney.fblayardsale.model.SaleItem;
import com.microsoft.windowsazure.mobileservices.MobileServiceList;
import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;

import java.util.ArrayList;

public class CommentListController {
    private static ArrayList<ItemComment> mComments = new ArrayList<>();
    private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();
    private static MobileServiceTable<ItemComment> mItemCommentTable;
    private static SaleItem mItem;

    public static void AttachAdapter(RecyclerView.Adapter adapter) {
        mAdapters.add(adapter);
    }

    public static int getCommentCount() {
        return mComments.size();
    }

    public static ItemComment getComment(int position) {
        if (mComments.size() > position) return mComments.get(position);
        else return null;
    }

    // Add a comment and notify the adapter of the change
    public static void addComment(ItemComment comment) {
        mComments.add(comment);
        mItem.setNumComments(mItem.getNumComments()+1);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
        // Update the count on the display, if shown
        MySalesController.notifyItem(mItem);
        LocalController.notifyItem(mItem);
    }

    // Remove a comment and notify the adapter of the change
    public static void removeComment(int position) {
        mComments.remove(position);
        mItem.setNumComments(mItem.getNumComments()-1);
        for (RecyclerView.Adapter adapter : mAdapters) {
            adapter.notifyDataSetChanged();
        }
        // Update the count on the display, if shown
        MySalesController.notifyItem(mItem);
        LocalController.notifyItem(mItem);
    }

    public static SaleItem getItem() { return mItem; }
    public static void setItem(SaleItem item) { mItem = item; }

    // Refresh all comments and notify the adapter of the change
    public static void Refresh(FblaAzure azure) {
        if (!azure.getLoggedOn()) return;
        mComments.clear();

        mItemCommentTable = azure.getClient().getTable(ItemComment.class);
        final MobileServiceTable<Account> mAccountTable = azure.getClient().getTable(
Account.class);
        new AsyncTask<Object, Object, Object>() {
            @Override
            protected Object doInBackground(Object... params) {
                try {
                    ArrayList<ItemComment> comments = new ArrayList<>();
                    final MobileServiceList<ItemComment> result =
                            mItemCommentTable.where().field("itemid").eq(mItem.getId()).
execute().get();
                    for (ItemComment comment : result) {
                        Account account = mAccountTable.lookUp(comment.getUserId()).get();
                        comment.setAccount(account);
                        comments.add(comment);
                    }
                    return comments;
                } catch (Exception exception) {
                    Log.e("CommentListController", exception.toString());
                }
                return null;
            }
            @Override
            protected void onPostExecute(Object result) {
                // If there are results, copy them into the array and notify the adapter.
                // This must be done on the UI thread.
                if (result != null) {
                    ArrayList<ItemComment> comments = (ArrayList<ItemComment>)result;
                    for (ItemComment comment : comments) {
                        mComments.add(comment);
                    }
                    mItem.setNumComments(comments.size());
                    for (RecyclerView.Adapter adapter : mAdapters) {
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        }.execute();
    }
}

```