File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\Help.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();
    }
}

```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\MySales.java

```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);
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\MySales.java

```
62                  break;
63              default:
64                  break;
65          }
66      }
67
68      @Override
69      public void onBackPressed()
70      {
71          this.finish();
72      }
73
74      @Override
75      public void onLogonComplete(Exception e) {
76          MySalesAdapter adapter = new MySalesAdapter(this, this, mAzure);
77          MySalesController.AttachAdapter(adapter);
78          mBinding.list.setAdapter(adapter);
79      }
80  }
81
```

```
/* 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();
```

```
62              return;
63          }
64
65          mAzure = new FblaAzure(this);
66          mAzure.setLogonListener(this);
67          mAzure.doLogon(userId, token);
68
69          mSaleItemTable = mAzure.getClient().getTable(SaleItem.class);
70
71          mBinding = DataBindingUtil.setContentView(this, R.layout.activity_addsales);
72          FblaPicture.setLayoutImage(mBinding.activityAddsales);
73          setSupportActionBar(mBinding.myToolbar);
74          mBinding.gallery.setOnClickListener(this);
75          mBinding.camera.setOnClickListener(this);
76          mBinding.back.setOnClickListener(this);
77          mBinding.finish.setOnClickListener(this);
78          mBinding.another.setOnClickListener(this);
79
80          mBinding.finish.setEnabled(false);
81          mBinding.another.setEnabled(false);
82
83          // Make sure Name is required.
84          mBinding.editname.addTextChangedListener(new TextWatcher() {
85              public void onTextChanged(CharSequence s, int start, int before, int count) {
86
87              }
88
89              public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {
90
91              }
92
93              public void afterTextChanged(Editable s) {
94                  if (s.length() > 0 && mBinding.editprice.getText().length() > 0) {
95                      mBinding.finish.setEnabled(true);
96                      mBinding.another.setEnabled(true);
97                  } else {
98                      mBinding.finish.setEnabled(false);
99                      mBinding.another.setEnabled(false);
100                 }
101             }
102         });
103
104         // Make sure Price is required.
105         mBinding.editprice.addTextChangedListener(new TextWatcher() {
106             public void onTextChanged(CharSequence s, int start, int before, int count) {
107
108             }
109
110             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {
111
112             }
113
114             public void afterTextChanged(Editable s) {
115                 if (s.length() > 0 && mBinding.editname.getText().length() > 0) {
116                     mBinding.finish.setEnabled(true);
117                     mBinding.another.setEnabled(true);
118                 } else {
119                     mBinding.finish.setEnabled(false);
120                     mBinding.another.setEnabled(false);
121                 }
122             }
```

```
123         });
124     }
125
126     @Override
127     public void onClick(View v) {
128         switch (v.getId()) {
129             case R.id.gallery:
130                 // Load a picture from the phone's gallery
131                 // Ask for permission first
132                 if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
133                     int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.
    permission.READ_EXTERNAL_STORAGE);
134                     if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
135                         // Should we show an explanation?
136                         if (ActivityCompat.shouldShowRequestPermissionRationale(this,
    Manifest.permission.READ_EXTERNAL_STORAGE)) {
137                             // Explain to the user why we need to read the contacts
138                         } else {
139                             ActivityCompat.requestPermissions(this,
140                                     new String[]{Manifest.permission.READ_EXTERNAL_STORAGE}
    , 0);
141                         }
142                         return;
143                     }
144                 }
145
146                 Intent i = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.
    Images.Media.EXTERNAL_CONTENT_URI);
147                 Log.d("CameraFragment", "Starting GALLERY Intent");
148                 this.startActivityForResult(i, 1);
149                 break;
150             case R.id.camera:
151                 // Take a picture from the camera.
152                 // Ask for permission first
153                 if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
154                     int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.
    permission.CAMERA);
155                     if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
156                         // Should we show an explanation?
157                         if (ActivityCompat.shouldShowRequestPermissionRationale(this,
    Manifest.permission.CAMERA)) {
158                             // Explain to the user why we need to read the contacts
159                         } else {
160                             ActivityCompat.requestPermissions(this,
161                                     new String[]{Manifest.permission.CAMERA}, 0);
162                         }
163                         return;
164                     }
165                 }
166
167                 Intent j = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
168                 this.startActivityForResult(j, 2);
169                 break;
170             case R.id.another:
171                 // Save the current item, and reload the activity to be ready for another
    one.
172                 addItem(v);
173                 this.finish();
174                 Intent it = new Intent(this, AddSales.class);
175                 Bundle b = new Bundle();
176                 b.putString("userId", mAzure.getUserId());
177                 b.putString("token", mAzure.getToken());
178                 it.putExtras(b);
```

```
179                 this.startActivity(it);
180                 break;
181             case R.id.finish:
182                 // Save the current item and return to YardSaleMain.
183                 addItem(v);
184                 this.finish();
185                 break;
186             default:
187                 // Don't save anything, just return to YardSaleMain.
188                 this.finish();
189                 break;
190         }
191     }
192
193     @Override
194     public void onBackPressed()
195     {
196         this.finish();
197     }
198
199     // Add a new item to the database.
200     private void addItem(View view) {
201         if (!mAzure.getLoggedOn()) return;
202
203         // Create a new item from the SaleItem model.
204         final SaleItem item = new SaleItem();
205         item.setId(UUID.randomUUID().toString());
206         item.setName(mBinding.editname.getText().toString());
207         item.setAccount(mAzure.getAccount());
208         item.setDescription(mBinding.editdesc.getText().toString());
209         String sPrice = mBinding.editprice.getText().toString();
210         if (sPrice == null || sPrice.equals("")) item.setPrice(0);
211         else item.setPrice(Float.parseFloat(mBinding.editprice.getText().toString()));
212         Bitmap b = FblaPicture.GetPictureFromView(mBinding.picture);
213         if (b != null) {
214             item.setPicture(b);
215         }
216         MySalesController.addItem(item);
217
218         // Save the item to the database over the internet.
219         AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
220             @Override
221             protected Void doInBackground(Void... params) {
222                 try {
223                     mSaleItemTable.insert(item);
224                     Log.d("AddSales:insert", "Created item " + item.getName());
225                     if (item.getHasPicture()) {
226                         Bitmap picture = item.getPicture();
227                         FblaPicture.UploadImage(item.getId(), picture);
228                         // For some strange reason, uploading the picture destroys it on the
    item. So put it back.
229                         item.setPicture(picture);
230                     }
231                     runOnUiThread(new Runnable() {
232                         @Override
233                         public void run() {
234                         }
235                     });
236                 } catch (Exception e) {
237                     Log.d("AddSales:insert", e.toString());
238                 }
239                 return null;
240             }
```

```
241         };
242         task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
243     }
244
245     @Override
246     public void onActivityResult(int requestCode, int resultCode, Intent data) {
247         // Results can come from the Camera, Gallery, or the Comments activity.
248         if (resultCode == android.app.Activity.RESULT_OK) {
249             if (requestCode == 2 && data != null) { // From Camera
250                 Log.d("AddSales", "Result from Camera");
251
252                 try {
253                     Bundle extras = data.getExtras();
254                     Bitmap image = (Bitmap) extras.get("data");
255                     image = FblaPicture.ResizePicture(this.getApplicationContext(), image);
256                     FblaPicture.LoadPictureOnView(mBinding.picture, image);
257                 } catch (Exception ex) {
258                     Log.e("AddSales:camera", ex.getMessage());
259                 }
260             } else if (requestCode == 1 && data != null) // Gallery
261             {
262                 // Gallery
263                 Log.d("AddSales", "Result from Gallery");
264
265                 try {
266                     Uri pickedImage = data.getData();
267                     InputStream stream = getContentResolver().openInputStream(pickedImage);
268                     Bitmap image = BitmapFactory.decodeStream(stream);
269                     image = FblaPicture.ResizePicture(this.getApplicationContext(), image);
270                     FblaPicture.LoadPictureOnView(mBinding.picture, image);
271                 } catch (Exception ex) {
272                     Log.e("AddSales:gallery", ex.getMessage());
273                 }
274             }
275         }
276     } // onActivityResult
277
278     @Override
279     public void onLogonComplete(Exception e) {
280
281     }
282 }
283
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\Comments.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");
    }

```

```
62      @Override
63      public void onClick(View v) {
64          switch (v.getId()) {
65              case R.id.post:
66                  if (!mBinding.newcomment.getText().toString().equals("")) {
67                      addItem(v);
68                  }
69                  break;
70              default:
71                  break;
72          }
73      }
74
75      @Override
76      public void onBackPressed()
77      {
78          this.finish();
79      }
80
81      // Add a new item to the database.
82      private void addItem(View view) {
83          if (!mAzure.getLoggedOn()) return;
84
85          // Create a new comment from the ItemComment model.
86          final ItemComment comment = new ItemComment();
87          comment.setId(UUID.randomUUID().toString());
88          comment.setComment(mBinding.newcomment.getText().toString());
89          comment.setUserId(mAzure.getUserId());
90          comment.setItemId(CommentListController.getItem().getId());
91          comment.setAccount(mAzure.getAccount());
92          CommentListController.addComment(comment);
93
94          // Save the item to the database over the internet.
95          AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
96              @Override
97              protected Void doInBackground(Void... params) {
98                  try {
99                      mCommentTable.insert(comment);
100                     Log.d("Comments:insert", "Created comment " + comment.getComment());
101                     runOnUiThread(new Runnable() {
102                         @Override
103                         public void run() {
104                         }
105                     });
106                 } catch (Exception e) {
107                     Log.d("Comments:insert", e.toString());
108                 }
109                 return null;
110             }
111         };
112         task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
113         if (this.getCurrentFocus() != null) {
114             InputMethodManager imm = (InputMethodManager) getSystemService(Context.
    INPUT_METHOD_SERVICE);
115             imm.hideSoftInputFromWindow(this.getCurrentFocus().getWindowToken(), 0);
116             mBinding.newcomment.setText("");
117         }
118     }
119
120     @Override
121     public void onLogonComplete(Exception e) {
122         CommentsAdapter adapter = new CommentsAdapter(this, this, mAzure);
123         CommentListController.AttachAdapter(adapter);
```

```
124         mBinding.list.setAdapter(adapter);
125     }
126 }
127
```

```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;

```

```
62      private Context mContext;
63      private ArrayList<LogonResultListener> mListeners = new ArrayList<LogonResultListener>(
    );
64
65      public FblaAzure (Context context) {
66          mContext = context;
67          try {
68              mClient = new MobileServiceClient(AZUREURL, mContext);
69          } catch (Exception e) {
70              Log.d("FblaAzure:init", e.toString());
71              mClient = null;
72          }
73      }
74
75      // It seems to use WebKit to perform the OAuth authentication via Azure.
76      // If successful, load the Account.  Otherwise return an exception to notify
77      // the listeners that it didn't work.
78      public void doLogon() {
79          Log.d("FblaAzure", "Logging on...");
80          ListenableFuture<MobileServiceUser> futureUser = mClient.login(PROVIDER);
81          Futures.addCallback(futureUser, new FutureCallback<MobileServiceUser>() {
82              @Override
83              public void onFailure(Throwable exc) {
84                  onLogonFailure((Exception)exc);
85              }
86              @Override
87              public void onSuccess(MobileServiceUser user) {
88                  mUser = user;
89                  doLoadAccount();
90              }
91          });
92      }
93
94      // This will load the account in the background.
95      public void doLoadAccount() {
96          new AsyncTask<Object, Object, Object>() {
97              @Override
98              protected Object doInBackground(Object... params) {
99                  Log.d("FblaAzure", "Loading Account...");
100                 return loadAccount();
101             }
102             @Override
103             protected void onPostExecute(Object result) {
104                 if (result == null) onLogonSuccess();
105                 else onLogonFailure((Exception)result);
106             }
107         }.execute();
108     }
109     public void doLogon(String userId, String token) {
110         mUser = new MobileServiceUser(userId);
111         mUser.setAuthenticationToken(token);
112         mClient.setCurrentUser(mUser);
113
114         new AsyncTask<Object, Object, Object>() {
115             @Override
116             protected Object doInBackground(Object... params) {
117                 Log.d("FblaAzure", "Loading Account...");
118                 return loadAccount();
119             }
120             @Override
121             protected void onPostExecute(Object result) {
122                 if (result == null) onLogonSuccess();
123                 else onLogonFailure((Exception)result);
```

```
124             }
125         }.execute();
126     }
127
128     public void doLogoff(YardSaleMain main) {
129         clearCookies();
130         mAccountTable = null;
131         mAccount = null;
132         mUser = null;
133         final YardSaleMain m = main;
134         ListenableFuture<MobileServiceUser> mLogout = mClient.logout();
135         Futures.addCallback(mLogout, new FutureCallback<MobileServiceUser>() {
136             @Override
137             public void onFailure(Throwable exc) {
138
139             }
140             @Override
141             public void onSuccess(MobileServiceUser user) {
142                 mClient = null;
143                 m.finish();
144             }
145         });
146         Log.d("FblaAzure:Logoff", "Logged Off");
147     }
148
149     public boolean getLoggedOn() { return mUser != null;}
150
151     public MobileServiceClient getClient() {
152         return mClient;
153     }
154
155     public String getUserId() {
156         if (mUser == null) return null;
157         else return mUser.getUserId();
158     }
159     public String getToken() {
160         if (mUser == null) return null;
161         else return mUser.getAuthenticationToken();
162     }
163
164     public Account getAccount() {
165         return mAccount;
166     }
167     public void setAccount(Account account) { mAccount = account; }
168
169     public int getSearchMiles(Context context) {
170         if (context == null) return 5;
171         SharedPreferences prefs = context.getSharedPreferences("settings", Context.
    MODE_PRIVATE);
172         if (prefs == null) return 5;
173         return prefs.getInt("miles", 5);
174     }
175
176     public void setSearchMiles(Context context, int miles) {
177         SharedPreferences prefs = context.getSharedPreferences("settings", Context.
    MODE_PRIVATE);
178         SharedPreferences.Editor editor = prefs.edit();
179         editor.putInt("miles", miles);
180         editor.commit();
181     }
182
183     // Once all of the logon processes are complete, notify any listeners
184     private void onLogonSuccess() {
```

```
185         Log.d("FblaAzure", "onLogonSuccess");
186         for (LogonResultListener listener : mListeners) {
187             listener.onLogonComplete(null);
188         }
189     }
190 
191     // Notify any listeners that the logon has failed.
192     private void onLogonFailure(Exception e) {
193         for (LogonResultListener listener : mListeners) {
194             listener.onLogonComplete(e);
195         }
196         Log.d("FblaAzure:Failure", e.toString());
197     }
198 
199     // It seems to use WebKit to perform the OAuth authentication via Azure.
200     // If successful, load the Account.  Otherwise return an exception to notify
201     // the listeners that it didn't work.
202     private Object providerLogon() {
203         try {
204             Log.d("FblaAzure:login", "Logging on to provider");
205             mUser = mClient.login(PROVIDER).get();
206             Log.d("FblaAzure:login", "Logged On");
207             return loadAccount();
208         } catch (Exception ex) {
209             Log.d("FblaAzure:login", ex.toString());
210             return ex;
211         }
212     }
213 
214     // A successfully loaded account means the token actually works and we can talk to Azure
    .
215     // Some accounts may not be linked to a School.
216     private Object loadAccount() {
217         // Now load the account
218         mAccountTable = mClient.getTable(Account.class);
219         try {
220             mAccount = mAccountTable.lookUp(mUser.getUserId()).get();
221             // Found the account record, so set it on the Data object.
222             Log.d("FblaAzure:account", "onSuccess - " + mAccount.getId());
223             return loadSchool();
224         } catch (ExecutionException e) {
225             if (e.getCause().getClass() == MobileServiceException.class) {
226                 MobileServiceException mEx = (MobileServiceException) e.getCause();
227                 if (mEx.getResponse() != null && mEx.getResponse().getStatus().code == 404)
    { // Not Found
228                     // The user is not in the table, so insert a new record for them.
229                     mAccount = new Account();
230                     mAccount.setId(mUser.getUserId());
231                     mAccountTable.insert(mAccount);
232                     Log.d("FblaAzure:account", "AccountEdit Created");
233                     return null;
234                 } else {
235                     Log.d("FblaAzure:account", mEx.toString());
236                     return mEx;
237                 }
238             } else {
239                 Log.d("FblaAzure:account", e.toString());
240                 return e;
241             }
242         } catch (Exception ex) {
243             // Something else bad happened.
244             Log.d("FblaAzure:account", ex.toString());
245             return ex;
```

```
246         }
247     }
248
249     // Load the school, if the Account is linked to one.
250     // Return a null if everything is successful. Otherwise return an Exception.
251     private Object loadSchool() {
252         if (mAccount.getSchoolId() != null && mAccount.getSchool() == null) {
253             Log.d("FblaAzure:school", "School " + mAccount.getSchoolId());
254             mSchoolsTable = mClient.getTable(Schools.class);
255             try {
256                 Schools school = mSchoolsTable.lookUp(mAccount.getSchoolId()).get();
257                 // Found the account record, so set it on the Data object.
258                 Log.d("FblaAzure:school", "onSuccess - "+school.getId());
259                 mAccount.setSchool(school);
260                 return null;
261             } catch (ExecutionException e) {
262                 if (e.getCause().getClass() == MobileServiceException.class) {
263                     MobileServiceException mEx = (MobileServiceException) e.getCause();
264                     if (mEx.getResponse() != null && mEx.getResponse().getStatus().code ==
    404) { // Not Found
265                         Log.d("FblaAzure:school", "School Missing");
266                         return null;
267                     } else {
268                         Log.d("FblaAzure:school", mEx.toString());
269                         return mEx;
270                     }
271                 } else {
272                     Log.d("FblaAzure:school", e.toString());
273                     return e;
274                 }
275             } catch (Exception ex) {
276                 // Something else bad happened.
277                 Log.d("FblaAzure:school", ex.toString());
278                 return ex;
279             }
280         } else {
281             Log.d("FblaAzure:school", "No School");
282             return null;
283         }
284     }
285
286     private void clearCookies() {
287         // Clear cookies and cache in order to logoff
288         CookieManager.getInstance().removeAllCookies(new ValueCallback<Boolean>() {
289             @Override
290             public void onReceiveValue(Boolean value) {
291                 Log.d("FblaAzure:cookies", "Cookies cleared");
292             }
293         });
294     }
295
296     // Add a listener to call after logon is complete
297     public void setLogonListener(LogonResultListener listener) {
298         mListeners.add(listener);
299     }
300
301     // This is the interface to use on the logon callbacks.
302     interface LogonResultListener {
303         void onLogonComplete(Exception e);
304     }
305 }
306
```

```
/* 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) {
```

```
63          super.onCreate(savedInstanceState);
64          setContentView(R.layout.activity_account);
65          mSchools = new ArrayList<Schools>(0);
66          Bundle b = getIntent().getExtras();
67          String userId = b.getString("userId");
68          String token = b.getString("token");
69          if (userId == null || token == null) {
70              Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
    LENGTH_LONG).show();
71              setResult(Activity.RESULT_CANCELED, new Intent());
72              finish();
73              return;
74          }
75          mBinding = DataBindingUtil.setContentView(this, R.layout.activity_account);
76          clearSchools();
77          mBinding.save.setEnabled(false);
78          setSupportActionBar(mBinding.myToolbar);
79
80          // Load the states onto the spinner from the resource file
81          mStateAdapter = ArrayAdapter.createFromResource(this, R.array.states_list, android.
    R.layout.simple_spinner_item);
82          mStateAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
    );
83          mBinding.state.setAdapter(mStateAdapter);
84
85          // Bind the schools array to the spinner
86          mSchoolAdapter = new ArrayAdapter<Schools>(this, android.R.layout.
    simple_spinner_item, mSchools);
87          mSchoolAdapter.setDropDownViewResource(android.R.layout.
    simple_spinner_dropdown_item);
88          mBinding.school.setAdapter(mSchoolAdapter);
89
90          mBinding.zip.setOnKeyListener(this);
91          mBinding.city.setOnKeyListener(this);
92          mBinding.save.setOnClickListener(this);
93          mBinding.cancel.setOnClickListener(this);
94          mBinding.searchZip.setOnClickListener(this);
95          mBinding.searchCityState.setOnClickListener(this);
96
97          // When you select a school, the city, state, and zip fields are updated with that
    school's info.
98          mBinding.school.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
    {
99              @Override
100             public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3)
     {
101                 Schools school = (Schools)mBinding.school.getSelectedItem();
102                 mBinding.zip.setText(school.getZip());
103                 mBinding.city.setText(school.getCity());
104                 int spinnerPosition = mStateAdapter.getPosition(school.getStateText());
105                 mBinding.state.setSelection(spinnerPosition, false);
106             }
107             @Override
108             public void onNothingSelected(AdapterView<?> arg0) {
109                 mBinding.zip.setText("");
110                 mBinding.city.setText("");
111                 mBinding.state.setSelection(0, false);
112
113             }
114         });
115
116         mAzure = new FblaAzure(this);
117         mAzure.setLogonListener(this);
```

```
118         mAzure.doLogon(userId, token);
119
120         int miles = mAzure.getSearchMiles(this);
121         switch (miles) {
122             case 5:
123                 mBinding.radius5.setChecked(true);
124                 break;
125             case 10:
126                 mBinding.radius10.setChecked(true);
127                 break;
128         }
129
130         // Make sure Name is required.
131         mBinding.name.addTextChangedListener(new TextWatcher() {
132             public void onTextChanged(CharSequence s, int start, int before, int count) {
133
134             }
135
136             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {
137
138             }
139
140             public void afterTextChanged(Editable s) {
141                 if (s.length() > 0) {
142                     mBinding.save.setEnabled(true);
143                 } else {
144                     mBinding.save.setEnabled(false);
145                 }
146             }
147         });
148     }
149
150     @Override
151     public void onClick(View v) {
152         // Clear the popup keyboard if it's there.
153         View view = this.getCurrentFocus();
154         if (view != null) {
155             InputMethodManager imm = (InputMethodManager)getSystemService(Context.
    INPUT_METHOD_SERVICE);
156             imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
157         }
158         // Decide which button was pressed.
159         switch (v.getId()) {
160             case R.id.save:
161                 // The radius value is not saved in the database. It's stored on the phone.
162                 int miles = mAzure.getSearchMiles(this);
163                 switch (miles) {
164                     case 5:
165                         if (mBinding.radius10.isChecked()) {
166                             mAzure.setSearchMiles(this, 10);
167                         }
168                         break;
169                     case 10:
170                         if (mBinding.radius5.isChecked()) {
171                             mAzure.setSearchMiles(this, 5);
172                         }
173                         break;
174                 }
175                 // Everything else is saved to the database on the Account
176                 if (mAzure.getLoggedOn()) {
177                     Account account = mAzure.getAccount();
178                     account.setName(mBinding.name.getText().toString());
```

```
179                     Object o = mBinding.school.getSelectedItem();
180                     if (o == null || ((Schools)o).getId() == "FAKE") {
181                         account.setSchool(null);
182                     }
183                     else {
184                         Schools school = (Schools)o;
185                         if (account.getSchoolId() == null || !account.getSchoolId().equals
    (school.getId())) {
186                             account.setSchool(school);
187                         }
188                     }
189 
190                     // The actual command to save to Azure must be done asynchronously
191                     AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object,
    Object>() {
192                         @Override
193                         protected Void doInBackground(Object... params) {
194                             try {
195                                 FblaAzure azure = (FblaAzure)params[0];
196                                 MobileServiceTable<Account> mAccountTable = azure.getClient
    ().getTable(Account.class);
197                                 Account account = azure.getAccount();
198                                 mAccountTable.update(account);
199                                 Log.d("AccountEdit:onClick", "AccountEdit Saved");
200                                 runOnUiThread(new Runnable() {
201                                     @Override
202                                     public void run() {
203                                         setResult(Activity.RESULT_OK, new Intent());
204                                         finish();
205                                     }
206                                 });
207                             } catch (Exception e) {
208                                 Log.d("AccountEdit", e.toString());
209                             }
210                             return null;
211                         }
212                     }.execute(mAzure);
213                 }
214 
215                 break;
216             case R.id.cancel:
217                 // This just closes the activity, returning you to YardSaleMain.
218                 setResult(Activity.RESULT_CANCELED, new Intent());
219                 this.finish();
220                 break;
221             case R.id.search_zip:
222                 // Executes a search for schools based on just the zip code.
223                 final String zipCode = mBinding.zip.getText().toString();
224                 if (zipCode.length() > 0) {
225                     clearSchools();
226                     searchSchools(zipCode);
227                 } else Log.d("AccountEdit:search", "FAIL");
228                 break;
229             case R.id.search_city_state:
230                 // Executes a search for schools based on city and state.
231                 int statePosition = mBinding.state.getSelectedItemPosition();
232                 if (statePosition > 0) {
233                     clearSchools();
234                     String stateSearch = mStateAdapter.getItem(statePosition).toString();
235                     String citySearch = mBinding.city.getText().toString();
236                     Log.d("AccountEdit:search", citySearch+", "+stateSearch);
237                     searchZip(citySearch, stateSearch);
238                 } else Log.d("AccountEdit:search", "FAIL");
```

```
239             default:
240                 break;
241         }
242     }
243
244     // This clears the schools spinner, but adds a fake "No School" entry
245     private void clearSchools() {
246         mSchools.clear();
247         Schools fake = new Schools();
248         fake.setId("FAKE");
249         fake.setSchool("No School Selected");
250         mSchools.add(fake);
251     }
252
253     // This removes the fake "No School" entry if it's there
254     private void addSchool(Schools school) {
255         if (mSchools.get(0).getId() == "FAKE") mSchools.remove(0);
256         mSchools.add(school);
257     }
258
259     // This is the actual search using city and state. It actually has to do two queries.
260     // First, we query the ZipCodes table for all of the zip codes matching the city and
    state.
261     // Then we get all of the schools in each zip code. Finally, we sort that list and load
262     // them into the array, which is bound to the spinner.
263     private void searchZip(final String city, final String state) {
264         AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object, Object>() {
265             @Override
266             protected Void doInBackground(Object... params) {
267                 try {
268                     FblaAzure azure = (FblaAzure)params[0];
269                     // Find matching zip codes for the city and state
270                     final MobileServiceList<ZipCodes> zipCodes =
271                             azure.getClient().getTable(ZipCodes.class).where()
272                                     .field("stateText").eq(state)
273                                     .and().startsWith("city", city)
274                                     .orderBy("city", QueryOrder.Ascending).execute().get();
275                     ArrayList<String> uniqueZips = new ArrayList<>();
276                     // Remove duplicate zip codes (some cities have multiple versions of the
     same name)
277                     for (ZipCodes zip : zipCodes) {
278                         if (!uniqueZips.contains(zip.getZip())) uniqueZips.add(zip.getZip()
    );
279                     }
280                     final ArrayList<Schools> allSchools = new ArrayList<>();
281                     for (String z : uniqueZips) {
282                         // Now get all of the schools in each zip code
283                         final MobileServiceList<Schools> schools =
284                                 azure.getClient().getTable(Schools.class).where()
285                                         .field("zip").eq(z)
286                                         .orderBy("school", QueryOrder.Ascending).execute().
    get();
287                         for (Schools school : schools) allSchools.add(school);
288                     }
289                     // Sort the list
290                     Collections.sort(allSchools);
291                     runOnUiThread(new Runnable() {
292                         @Override
293                         public void run() {
294                         // Put the schools into the array and notify the spinner that it's
    different.
295                         for (Schools s : allSchools) {
296                             addSchool(s);
```

```
297                         }
298                         mSchoolAdapter.notifyDataSetChanged();
299                         }
300                     });
301
302                 } catch (Exception e) {
303                     Log.d("SearchZip", e.toString());
304                 }
305                 return null;
306             }
307         }.execute(mAzure);
308     }
309
310     // This is a very straight forward fetch of all schools in a given zip code.
311     private void searchSchools(final String zip) {
312         AsyncTask<Object, Object, Object> task = new AsyncTask<Object, Object, Object>() {
313             @Override
314             protected Void doInBackground(Object... params) {
315                 try {
316                     FblaAzure azure = (FblaAzure)params[0];
317                     final MobileServiceList<Schools> schools =
318                             azure.getClient().getTable(Schools.class).where()
319                                     .field("zip").eq(zip)
320                                     .orderBy("school", QueryOrder.Ascending).execute().get(
    );
321                     runOnUiThread(new Runnable() {
322                         @Override
323                         public void run() {
324                             for (Schools s : schools) {
325                                 addSchool(s);
326                                 Log.d("SearchSchool", s.toString());
327                             }
328                             mSchoolAdapter.notifyDataSetChanged();
329                         }
330                     });
331
332                 } catch (Exception e) {
333                     Log.d("SearchSchool", e.toString());
334                 }
335
336                 return null;
337             }
338         }.execute(mAzure);
339     }
340
341     @Override
342     public void onBackPressed() {
343         setResult(Activity.RESULT_CANCELED, new Intent());
344         this.finish();
345     }
346
347     // This allows the user to press enter on the popup keyboard and
348     // have it automatically execute the corresponding search (zip or city/state).
349     @Override
350     public boolean onKey(View v, int keyCode, KeyEvent event) {
351         if (event.getAction() == KeyEvent.ACTION_DOWN)
352         {
353             switch (keyCode)
354             {
355                 case KeyEvent.KEYCODE_DPAD_CENTER:
356                 case KeyEvent.KEYCODE_ENTER:
357                     switch (v.getId()) {
358                         case R.id.zip:
```

```
359                             this.onClick(mBinding.searchZip);
360                             break;
361                         case R.id.city:
362                             this.onClick(mBinding.searchCityState);
363                             break;
364                     }
365                     return true;
366                 default:
367                     break;
368             }
369         }
370         return false;
371     }
372
373     @Override
374     public void onLogonComplete(Exception e) {
375         // Display your current school.
376         Account account = mAzure.getAccount();
377         mBinding.name.setText(account.getName());
378         mBinding.save.setEnabled(account.getName().length() > 0);
379         if (account.getSchool() != null) {
380             Schools school = account.getSchool();
381             addSchool(school);
382             mSchoolAdapter.notifyDataSetChanged();
383             mBinding.zip.setText(school.getZip());
384             mBinding.city.setText(school.getCity());
385             int spinnerPosition = mStateAdapter.getPosition(school.getStateText());
386             mBinding.state.setSelection(spinnerPosition, false);        }
387     }
388 }
389
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\FblaPicture.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=
Tjly1N1KDieodg23eAAgq0bV6rvLpxUM3PAAGGkjWp5Jf8XshGhr87agsbWMrYyEwgMTQ4MhoeK7L4kxdv9Agg==;
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
```

```
61      public static Point GetSize(Context c)
62      {
63          WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
64          Display display = wm.getDefaultDisplay();
65          Point size = new Point();
66          if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.
    HONEYCOMB_MR2)
67          {
68              display.getSize(size);
69          }
70          else // Old Version
71          {
72              size.set(display.getWidth(), display.getHeight());
73          }
74          return size;
75      }
76
77      // Loads a bitmap picture onto the ImageView item on the layout.
78      public static void LoadPictureOnView(ImageView view, Bitmap original) {
79          int vh = getImageHeight();
80          view.setMinimumHeight(vh);
81          view.setMaxHeight(vh);
82          view.setImageBitmap(original);
83      }
84
85      public static Bitmap GetPictureFromView(ImageView view) {
86          Drawable d = view.getDrawable();
87          if (d == null) return null;
88          return ((BitmapDrawable)d).getBitmap();
89      }
90
91      // Resizes a picture selected from the gallery or taken by the camera so they are a
    common size.
92      public static Bitmap ResizePicture(Context c, Bitmap original) {
93          int w = original.getWidth();
94          int h = original.getHeight();
95          Point screen = GetSize(c);
96          // Force everything to be 500 pixels long
97          int screenL = 500;
98          int originL = (w > h) ? w : h;
99          int originS = (w > h) ? h : w;
100
101         int newS = (int)((float)screenL * ((float)originS / (float)originL));
102         if (w > h)
103         {
104             Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
     + w + "x" + h + " to " + screenL + "x" + newS);
105             return Bitmap.createScaledBitmap(original, screenL, newS, true);
106         }
107         else
108         {
109             Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
     + w + "x" + h + " to " + newS + "x" + screenL);
110             return Bitmap.createScaledBitmap(original, newS, screenL, true);
111         }
112     }
113
114     // Uploads a picture to Azure storage. This must be run in the background.
115     public static void UploadImage(String itemId, Bitmap image) throws URISyntaxException,
    InvalidKeyException, StorageException, IOException {
116         ByteArrayOutputStream baos = new ByteArrayOutputStream();
117         image.compress(Bitmap.CompressFormat.PNG, 100, baos);
118         byte[] b = baos.toByteArray();
```

```
119         ByteArrayInputStream bs = new ByteArrayInputStream(b);
120         synchronized (containerLock) {
121             if (mContainer == null) {
122                 CloudStorageAccount storageAccount = CloudStorageAccount.parse(
    mStorageConnection);
123                 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
124                 mContainer = blobClient.getContainerReference("yardsale");
125             }
126             CloudBlockBlob imageBlob = mContainer.getBlockBlobReference(itemId);
127             imageBlob.upload(bs, b.length);
128             Log.d("UploadImage", "Uploaded " + itemId);
129         }
130     }
131
132     // Downloads a picture from Azure storage. This must be run in the background.
133     // If the image does not exist, it will return a null.
134     public static Bitmap DownloadImage(String itemId) {
135         try {
136             ByteArrayOutputStream bs = new ByteArrayOutputStream();
137             synchronized (containerLock) {
138                 if (mContainer == null) {
139                     CloudStorageAccount storageAccount = CloudStorageAccount.parse(
    mStorageConnection);
140                     CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
141                     mContainer = blobClient.getContainerReference("yardsale");
142                 }
143                 CloudBlockBlob blob = mContainer.getBlockBlobReference(itemId);
144                 if (blob.exists()) {
145                     blob.download(bs);
146                 } else {
147                     return null;
148                 }
149             }
150             // Convert to bitmap
151             byte[] b = bs.toByteArray();
152             return BitmapFactory.decodeByteArray(b, 0, b.length);
153         } catch (Exception ex) {
154             Log.d("FblaPicture:Download", itemId + " - " + ex.toString());
155         }
156         return null;
157     }
158
159     public static void DeleteImage(String itemId) {
160         try {
161             synchronized (containerLock) {
162                 if (mContainer == null) {
163                     CloudStorageAccount storageAccount = CloudStorageAccount.parse(
    mStorageConnection);
164                     CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
165                     mContainer = blobClient.getContainerReference("yardsale");
166                 }
167                 CloudBlockBlob blob = mContainer.getBlockBlobReference(itemId);
168                 if (blob.exists()) {
169                     blob.delete();
170                 }
171             }
172         } catch (Exception ex) {
173             Log.d("FblaPicture:Delete", itemId + " - " + ex.toString());
174         }
175     }
176 }
177
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\HelpAdapter.java

```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;
```

```
64              case 9:
65                  v = LayoutInflater.from(parent.getContext())
66                          .inflate(R.layout.help10_numbersales, parent, false);
67                  break;
68              case 10:
69                  v = LayoutInflater.from(parent.getContext())
70                          .inflate(R.layout.help11_logout, parent, false);
71                  break;
72              default:
73                  v = null;
74                  break;
75          }
76          if (v == null) return null;
77          return new HelpAdapter.ViewHolder(v);
78      }
79
80      @Override
81      public void onBindViewHolder(HelpAdapter.ViewHolder holder, int position) {
82
83      }
84
85      @Override
86      public int getItemCount() {
87          return 11;
88      }
89
90      class ViewHolder extends RecyclerView.ViewHolder {
91
92          public ViewHolder(View itemView) {
93              super(itemView);
94          }
95      }
96  }
97
```

```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);
```

```
64          return fragment;
65      }
66
67      public MapFragment() {
68          // Required empty public constructor
69      }
70
71      public void setEnabled(boolean enable) {
72
73      }
74
75      @Override
76      public void onAttach(Context context) {
77          super.onAttach(context);
78          try {
79              mListener = (MapFragment.OnFragmentInteractionListener) context;
80              mListener.onMapAttach(this);
81              LocalController.attachRefreshListener(this);
82          } catch (ClassCastException e) {
83              throw new ClassCastException(context.toString()
84                      + " must implement OnFragmentInteractionListener");
85          }
86      }
87
88      @Override
89      public void onDetach() {
90          super.onDetach();
91          mListener.onMapDetach(this);
92          mListener = null;
93          LocalController.detachRefreshListener(this);
94      }
95
96      @Override
97      public void onCreate(Bundle savedInstanceState) {
98          super.onCreate(savedInstanceState);
99      }
100
101     @Override
102     public View onCreateView(LayoutInflater inflater, ViewGroup container,
103                              Bundle savedInstanceState) {
104
105         mBinding = DataBindingUtil.inflate(
106                 inflater, R.layout.fragment_map, container, false);
107         View view = mBinding.getRoot();
108
109         Log.d("Map:onCreateView", "Start");
110         mBinding.map.onCreate(savedInstanceState);
111         mBinding.map.onResume();
112
113         try {
114             MapsInitializer.initialize(getActivity().getApplicationContext());
115         } catch (Exception e) {
116             Log.d("Map:onCreateView", e.getMessage());
117         }
118
119         mBinding.map.getMapAsync(new OnMapReadyCallback() {
120             @Override
121             public void onMapReady(GoogleMap googleMap) {
122                 mMap = googleMap;
123                 Log.d("Map", "Ready");
124                 loadMap();
125             }
126         });
```

```
127
128         return view;
129     }
130
131     private void loadMap() {
132         if (mMap == null) return;
133         mMap.clear();
134         YardSaleMain main = (YardSaleMain)getActivity();
135         FblaAzure azure = main.getAzure();
136         mMap.getUiSettings().setZoomControlsEnabled(true);
137         mMap.getUiSettings().setAllGesturesEnabled(true);
138
139         if (azure.getAccount() == null) return;
140         Schools mySchool = azure.getAccount().getSchool();
141         if (mySchool == null) return;
142         // Get all of the distinct schools from the LocalController
143         ArrayList<Schools> schools = new ArrayList<>();
144         HashMap<String, Integer> counts = new HashMap<String, Integer>();
145         for (int i = 0; i < LocalController.getItemCount(); i++) {
146             SaleItem item = LocalController.getItem(i);
147             Schools school = item.getAccount().getSchool();
148             if (!schools.contains(school)) {
149                 schools.add(school);
150                 counts.put(school.getId(), new Integer(1));
151             } else {
152                 counts.put(school.getId(), counts.get(school.getId()) + 1);
153             }
154         }
155         LatLng myLL = null;
156         // Mark all nearby schools on the map.
157         for (Schools s : schools) {
158             String title = s.getSchool();
159             Integer sales = counts.get(s.getId());
160             String desc = "Items for sale: " + sales.toString();
161             LatLng ll = new LatLng(s.getLat(), s.getLong());
162             BitmapDescriptor bm;
163             if (s.getId().equals(mySchool.getId())) {
164                 myLL = ll;
165                 bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
    HUE_AZURE);
166             } else {
167                 // Change the hue of the marker depending on how many items are for sale at
     the school.
168                 switch (sales) {
169                     case 1:
170                     case 2:
171                         bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
    HUE_YELLOW);
172                         break;
173                     case 3:
174                     case 4:
175                         bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
    HUE_ORANGE);
176                         break;
177                     default:
178                         bm = BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
    HUE_RED);
179                         break;
180                 }
181             }
182             mMap.addMarker(new MarkerOptions().position(ll)
183                     .title(title).snippet(desc).icon(bm));
184         }
```

```
185         // See if your school has already been marked. If not, mark it.
186         if (myLL == null) {
187             myLL = new LatLng(mySchool.getLat(), mySchool.getLong());
188             String title = mySchool.getSchool();
189             String desc = "Items for sale: 0";
190             mMap.addMarker(new MarkerOptions().position(myLL)
191                     .title(title).snippet(desc)
192                     .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
    HUE_AZURE)));
193         }
194         // Set the camera on your school
195         mMap.moveCamera(CameraUpdateFactory.newLatLng(myLL));
196         mMap.animateCamera(CameraUpdateFactory.zoomTo(12.0f));
197     }
198
199     @Override
200     public void onResume() {
201         super.onResume();
202         Log.d("Map", "Resume");
203         loadMap();
204         mBinding.map.onResume();
205     }
206
207     @Override
208     public void onPause() {
209         super.onPause();
210         mBinding.map.onPause();
211     }
212
213     @Override
214     public void onDestroy() {
215         super.onDestroy();
216         mBinding.map.onDestroy();
217     }
218
219     @Override
220     public void onLowMemory() {
221         super.onLowMemory();
222         mBinding.map.onLowMemory();
223     }
224
225     // Initializes layout items
226     @Override
227     public void onActivityCreated(Bundle bundle) {
228         super.onActivityCreated(bundle);
229     }
230
231 }
232
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\HomeFragment.java

```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);
```

```
64                      getActivity().startActivity(i);
65                  }
66                  break;
67              case R.id.help:
68                  getActivity().startActivity(new Intent(getActivity(), Help.class));
69                  break;
70              case R.id.logout:
71                  // Logout is a problem. Azure doesn't seem to be able to handle it
72                  // when I clear the cookies in order to force a new logon prompt.
73                  // If you try running the app too soon after logging off,
74                  // you get this strange net::ERR_EMPTY_RESPONSE error, which is coming
75                  // from the Azure library itself. Because of that problem, we are removing
76                  // the logoff feature and relabeling this button "Close App"
77                  //main.Logoff();
78                  getActivity().finish();
79                  break;
80              default:
81                  break;
82          }
83      }
84
85      public void setEnabled(boolean enable) {
86          if (mBinding != null)
87              mBinding.fragmentHome.setEnabled(enable);
88      }
89
90      public interface OnFragmentInteractionListener {
91          public void onHomeAttach(HomeFragment f);
92          public void onHomeDetach(HomeFragment f);
93      }
94
95      // Implementation of Fragment
96      public static HomeFragment newInstance(String param1, String param2) {
97          HomeFragment fragment = new HomeFragment();
98          Bundle args = new Bundle();
99          fragment.setArguments(args);
100         return fragment;
101     }
102
103     public HomeFragment() {
104         // Required empty public constructor
105     }
106
107     @Override
108     public void onAttach(Context context) {
109         super.onAttach(context);
110         try {
111             mListener = (OnFragmentInteractionListener) context;
112             mListener.onHomeAttach(this);
113         } catch (ClassCastException e) {
114             throw new ClassCastException(context.toString()
115                     + " must implement OnFragmentInteractionListener");
116         }
117     }
118
119     @Override
120     public void onDetach() {
121         super.onDetach();
122         mListener.onHomeDetach(this);
123         mListener = null;
124     }
125
126     @Override
```

```
127     public void onCreate(Bundle savedInstanceState) {
128         super.onCreate(savedInstanceState);
129     }
130
131     @Override
132     public View onCreateView(LayoutInflater inflater, ViewGroup container,
133                              Bundle savedInstanceState) {
134
135         mBinding = DataBindingUtil.inflate(
136                 inflater, R.layout.fragment_home, container, false);
137         mBinding.account.setOnClickListener(this);
138         mBinding.add.setOnClickListener(this);
139         mBinding.my.setOnClickListener(this);
140         mBinding.help.setOnClickListener(this);
141         mBinding.logout.setOnClickListener(this);
142         View view = mBinding.getRoot();
143
144         return view;
145     }
146
147     // Initializes layout items
148     @Override
149     public void onActivityCreated(Bundle bundle) {
150         super.onActivityCreated(bundle);
151     }
152
153 }
154
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\LocalAdapter.java

```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) {
```

```
62                  mBinding.user.setText(account.getName());
63                  Schools school = account.getSchool();
64                  if (school != null) {
65                      mBinding.address.setText(school.getFullAddress());
66                      mBinding.chapter.setText(school.getSchool());
67                  }
68              }
69              Bitmap image = item.getPicture();
70              if (image != null) {
71                  FblaPicture.setLayoutImage(mBinding.layoutPicture);
72                  FblaPicture.LoadPictureOnView(mBinding.picture, image);
73              }
74          }
75      }
76
77      @Override
78      public int getItemCount() {
79          return LocalController.getItemCount();
80      }
81
82      @Override
83      public void onClick(View v) {
84          switch (v.getId()) {
85              case R.id.comments:
86                  if (mAzure.getLoggedOn()) {
87                      int position = (int)v.getTag();
88                      CommentListController.setItem(LocalController.getItem(position));
89                      CommentListController.Refresh(mAzure);
90                      mParentListener.onClick(v);
91                  }
92                  break;
93          }
94      }
95
96      public class ViewHolder extends RecyclerView.ViewHolder {
97          private ListItemsBinding mBinding;
98
99          public ViewHolder(View itemView, ListItemsBinding binding) {
100             super(itemView);
101             mBinding = binding;
102         }
103
104         public ListItemsBinding getBinding() {
105             return mBinding;
106         }
107     }
108 }
109
```

```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);
```

```
62          mPagerAdapter.onPageSelected(0);
63          mBinding.home.setOnClickListener(this);
64          mBinding.local.setOnClickListener(this);
65          mBinding.map.setOnClickListener(this);
66          setSupportActionBar(mBinding.myToolbar);
67          mTitle = mBinding.myToolbar.getTitle().toString();
68
69          // Make sure everything is disabled until the logon completes
70          mBinding.local.setEnabled(false);
71          mBinding.map.setEnabled(false);
72          mBinding.pager.setEnabled(false);
73
74          mAzure.setLogonListener(this);
75          mAzure.doLogon();
76      }
77
78      //@Override
79      public void onClick(View v) {
80          // Perform page changes so they transition just like a swipe.
81          int pg;
82          switch (v.getId()) {
83              case R.id.home:
84                  pg = 0;
85                  break;
86              case R.id.local:
87                  pg = 1;
88                  break;
89              default:
90                  pg = 2;
91                  break;
92          }
93          mBinding.pager.setCurrentItem(pg, true);
94      }
95
96      public FblaAzure getAzure() { return mAzure; }
97
98      private HomeFragment mHomeFragment = null;
99      public void onHomeAttach(HomeFragment f) {
100         mHomeFragment = f;
101         mHomeFragment.setEnabled(mLogonComplete);
102     }
103     public void onHomeDetach(HomeFragment f) {
104         mHomeFragment = null;
105     }
106
107     private LocalFragment mLocalFragment = null;
108     public void onLocalAttach(LocalFragment f) {
109         mLocalFragment = f;
110         mLocalFragment.setEnabled(mLogonComplete);
111     }
112     public void onLocalDetach(LocalFragment f) {
113         mLocalFragment = null;
114     }
115
116     private MapFragment mMapFragment = null;
117     public void onMapAttach(MapFragment f) {
118         mMapFragment = f;
119         mMapFragment.setEnabled(mLogonComplete);
120     }
121     public void onMapDetach(MapFragment f) {
122         mMapFragment = null;
123     }
124
```

```
125     public void onLogonComplete(Exception e) {
126         if (e != null) {
127             Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
    LENGTH_LONG).show();
128             finish();
129         } else if (!mLogonComplete) {
130             Log.d("YardSaleMain", "Logon Complete");
131             Account account = mAzure.getAccount();
132             mLogonComplete = true;
133             mBinding.myToolbar.setTitle(mTitle + " - " + account.getName());
134             mBinding.local.setEnabled(true);
135             mBinding.map.setEnabled(true);
136             mBinding.pager.setEnabled(true);
137             if (mHomeFragment != null) {
138                 mHomeFragment.setEnabled(true);
139             }
140             if (mLocalFragment != null) {
141                 mLocalFragment.setEnabled(true);
142             }
143             if (mMapFragment != null) {
144                 mMapFragment.setEnabled(true);
145             }
146
147             if (account.getName() == null || account.getName().equals("")) {
148                 Intent i = new Intent(this, AccountEdit.class);
149                 Bundle b = new Bundle();
150                 b.putString("userId", mAzure.getUserId());
151                 b.putString("token", mAzure.getToken());
152                 i.putExtras(b);
153                 this.startActivityForResult(i, 0);
154             } else {
155                 MySalesController.Refresh(mAzure);
156                 LocalController.Refresh(this, mAzure);
157             }
158         } else {
159             Account account = mAzure.getAccount();
160             MySalesController.Refresh(mAzure);
161             LocalController.Refresh(this, mAzure);
162             mBinding.myToolbar.setTitle(mTitle + " - " + account.getName());
163         }
164     }
165
166     // Require two presses on the back button to exit the activity.
167     private Boolean exit = false;
168     @Override
169     public void onBackPressed() {
170         if (exit) {
171             finish(); // finish activity
172         } else {
173             Toast.makeText(this, "Press Back again to Exit.",
174                     Toast.LENGTH_SHORT).show();
175             exit = true;
176             new Handler().postDelayed(new Runnable() {
177                 @Override
178                 public void run() {
179                     exit = false;
180                 }
181             }, 3 * 1000);
182
183         }
184
185     }
186
```

```
187     @Override
188     public void onActivityResult(int requestCode, int resultCode, Intent data) {
189         super.onActivityResult(requestCode, resultCode, data);
190         if (resultCode == RESULT_OK) {
191             mAzure.doLoadAccount();
192         }
193     }
194
195     @Override
196     public void onDestroy() {
197         super.onDestroy();
198         mAzure = null;
199     }
200 }
201
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\LocalFragment.java

```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);
    }
```

```
64
65      // Implementation of Fragment
66      public static LocalFragment newInstance(String param1, String param2) {
67          LocalFragment fragment = new LocalFragment();
68          Bundle args = new Bundle();
69          fragment.setArguments(args);
70          return fragment;
71      }
72
73      public LocalFragment() {
74          // Required empty public constructor
75      }
76
77      @Override
78      public void onAttach(Context context) {
79          super.onAttach(context);
80          try {
81              mListener = (LocalFragment.OnFragmentInteractionListener) context;
82              mListener.onLocalAttach(this);
83          } catch (ClassCastException e) {
84              throw new ClassCastException(context.toString()
85                      + " must implement OnFragmentInteractionListener");
86          }
87      }
88
89      @Override
90      public void onDetach() {
91          super.onDetach();
92          mListener.onLocalDetach(this);
93          mListener = null;
94      }
95
96      @Override
97      public void onCreate(Bundle savedInstanceState) {
98          super.onCreate(savedInstanceState);
99      }
100
101     @Override
102     public View onCreateView(LayoutInflater inflater, ViewGroup container,
103                              Bundle savedInstanceState) {
104         mBinding = DataBindingUtil.inflate(
105                 inflater, R.layout.fragment_local, container, false);
106         View view = mBinding.getRoot();
107         return view;
108     }
109
110     @Override
111     public void onActivityCreated(Bundle bundle) {
112         super.onActivityCreated(bundle);
113         // Setup the RecyclerView here because the data changes.
114         YardSaleMain mParent = (YardSaleMain)getActivity();
115         mBinding.list.setLayoutManager(new LinearLayoutManager(mParent));
116         LocalAdapter adapter = new LocalAdapter(this, mParent.getAzure());
117         LocalController.AttachAdapter(adapter);
118         LocalController.Refresh(mParent, mParent.getAzure());
119         mBinding.list.setAdapter(adapter);
120     }
121
122 }
123
```

```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();
```

```
61              mBinding.comments.setTag(position);
62              mBinding.name.setText(item.getName());
63              mBinding.price.setText(String.format("$%.2f", item.getPrice()));
64              mBinding.description.setText(item.getDescription());
65              mBinding.comments.setText("COMMENTS (" + item.getNumComments() + ")");
66              mBinding.sold.setTag(position); // Being sold means to delete it.
67              if (item.getHasPicture()) {
68                  Bitmap image = item.getPicture();
69                  FblaPicture.setLayoutImage(mBinding.layoutPicture);
70                  FblaPicture.LoadPictureOnView(mBinding.picture, image);
71              }
72          }
73      }
74
75      @Override
76      public int getItemCount() {
77          return MySalesController.getItemCount();
78      }
79
80      @Override
81      public void onClick(View v) {
82          switch (v.getId()) {
83              case R.id.comments:
84                  if (mAzure.getLoggedOn()) {
85                      int position = (int)v.getTag();
86                      SaleItem item = MySalesController.getItem(position);
87                      CommentListController.setItem(item);
88                      CommentListController.Refresh(mAzure);
89                      Log.d("MySalesAdapter", "Refreshed for " + position);
90                      mParentListener.onClick(v);
91                  }
92                  break;
93              case R.id.sold:
94                  // When it's sold, we just delete it.
95                  final int position = (int)v.getTag();
96                  AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
97                  builder.setTitle("Are You Sure?");
98                  final TextView info = new TextView(mContext);
99                  info.setText("By Pressing Confirm, The Item Will Be Deleted.");
100                 info.setPadding(30, 0, 0, 0);
101                 builder.setView(info);
102
103                 builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener()
    {
104                     @Override
105                     public void onClick(DialogInterface dialog, int which) {
106                         Log.d("MySalesAdapter", "delete");
107                         deleteItem(position);
108                         dialog.dismiss();
109                     }
110                 });
111
112                 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
113                     @Override
114                     public void onClick(DialogInterface dialog, int which) {
115                         dialog.cancel();
116                     }
117                 });
118
119                 builder.show();
120                 break;
121             default:
122                 break;
```

```
123         }
124     }
125
126     public class ViewHolder extends RecyclerView.ViewHolder {
127         private ListItemsBinding mBinding;
128
129         public ViewHolder(View itemView, ListItemsBinding binding) {
130             super(itemView);
131             mBinding = binding;
132         }
133
134         public ListItemsBinding getBinding() {
135             return mBinding;
136         }
137     }
138
139     private void deleteItem(int position) {
140         if (!mAzure.getLoggedOn()) return;
141
142         final int pos = position;
143         final SaleItem item = MySalesController.getItem(position);
144         final MobileServiceTable<SaleItem> mSaleItemTable = mAzure.getClient().getTable(
    SaleItem.class);
145         // Delete the comment from the database.
146         AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
147             @Override
148             protected Void doInBackground(Void... params) {
149                 try {
150                     FblaPicture.DeleteImage(item.getId());
151                     mSaleItemTable.delete(item);
152                     Log.d("MySales:delete", "Deleted item " + item.getName());
153                     mContext.runOnUiThread(new Runnable() {
154                         @Override
155                         public void run() {
156                             MySalesController.removeItem(pos);
157                         }
158                     });
159                 } catch (Exception e) {
160                     Log.d("MySales:delete", e.toString());
161                 }
162                 return null;
163             }
164         };
165         task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
166     }
167 }
168
```

```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());
```

```
61              if (comment.getAccount() == null) mBinding.username.setText("{Unknown}");
62              else mBinding.username.setText(comment.getAccount().getName());
63              mBinding.delete.setTag(position);
64          }
65      }
66
67      @Override
68      public int getItemCount() {
69          return CommentListController.getCommentCount();
70      }
71
72      @Override
73      public void onClick(View v) {
74          switch (v.getId()) {
75              case R.id.delete:
76                  final int position = (int)v.getTag();
77                  AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
78                  builder.setTitle("Are You Sure?");
79                  final TextView info = new TextView(mContext);
80                  info.setText("By Pressing Confirm, The Comment Will Be Deleted.");
81                  info.setPadding(30, 0, 0, 0);
82                  builder.setView(info);
83
84                  builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener()
    {
85                      @Override
86                      public void onClick(DialogInterface dialog, int which) {
87                          deleteComment(position);
88                          dialog.dismiss();
89                      }
90                  });
91
92                  builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
93                      @Override
94                      public void onClick(DialogInterface dialog, int which) {
95                          dialog.cancel();
96                      }
97                  });
98
99                  builder.show();
100                 break;
101             default:
102                 break;
103         }
104     }
105
106     private void deleteComment(int position) {
107         if (!mAzure.getLoggedOn()) return;
108
109         final int pos = position;
110         final ItemComment comment = CommentListController.getComment(position);
111         final MobileServiceTable<ItemComment> mCommentTable = mAzure.getClient().getTable(
    ItemComment.class);
112         // Delete the comment from the database.
113         AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
114             @Override
115             protected Void doInBackground(Void... params) {
116                 try {
117                     mCommentTable.delete(comment);
118                     Log.d("Comments:delete", "Deleted comment " + comment.getComment());
119                     mContext.runOnUiThread(new Runnable() {
120                         @Override
121                         public void run() {
```

```
122                             CommentListController.removeComment(pos);
123                         }
124                     });
125                 } catch (Exception e) {
126                     Log.d("Comments:delete", e.toString());
127                 }
128                 return null;
129             }
130         };
131         task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
132     }
133
134     public class ViewHolder extends RecyclerView.ViewHolder {
135         private ListCommentsBinding mBinding;
136
137         public ViewHolder(View itemView, ListCommentsBinding binding) {
138             super(itemView);
139             mBinding = binding;
140         }
141
142         public ListCommentsBinding getBinding() {
143             return mBinding;
144         }
145     }
146 }
147
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\FblaPagerAdapter.java

```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)
  {

```

```
62      }
63
64      @Override
65      public void onPageSelected(int position) {
66          switch (position) {
67              case 0: // Home
68                  mContext.mBinding.home.setTextColor(Color.BLACK);
69                  mContext.mBinding.local.setTextColor(Color.DKGRAY);
70                  mContext.mBinding.map.setTextColor(Color.DKGRAY);
71                  break;
72              case 1: // Local
73                  mContext.mBinding.home.setTextColor(Color.DKGRAY);
74                  mContext.mBinding.local.setTextColor(Color.BLACK);
75                  mContext.mBinding.map.setTextColor(Color.DKGRAY);
76                  break;
77              case 2: // Map
78                  mContext.mBinding.home.setTextColor(Color.DKGRAY);
79                  mContext.mBinding.local.setTextColor(Color.DKGRAY);
80                  mContext.mBinding.map.setTextColor(Color.BLACK);
81                  break;
82              default:
83                  Log.d("FblaPager:Selected", "Other");
84                  break;
85          }
86      }
87
88      @Override
89      public void onPageScrollStateChanged(int state) {
90
91      }
92  }
93
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\model\Account.java

```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; }
```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\model\Schools.java

```
63      public final void setSchool(String school) { mSchool = school; }
64      public String getAddress() { return mAddress; }
65      public final void setAddress(String address) { mAddress = address; }
66      public String getCity() { return mCity; }
67      public final void setCity(String city) { mCity = city; }
68      public String getStateText() { return mStateText; }
69      public final void setStateText(String stateText) { mStateText = stateText; }
70      public double getLat() { return mLat; }
71      public final void setLat(double lat) { mLat = lat; }
72      public double getLong() { return mLong; }
73      public final void setLong(double lng) { mLong = lng; }
74
75      // Separate full address for displaying with an item.
76      public String getFullAddress() {
77          return mAddress + ", " + mCity + ", " + mStateText;
78      }
79
80      @Override
81      public boolean equals(Object o) {
82          return o instanceof Schools && ((Schools)o).mId == mId;
83      }
84
85      // Implements Comparable so we can sort them during a city/state search.
86      @Override
87      public int compareTo(@NonNull Schools o) {
88          return this.getSchool().compareTo(o.getSchool());
89      }
90  }
91
```

```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; }
```

```
64      public String getDescription() { return mDescription; }
65      public final void setDescription(String description) { mDescription = description; }
66      public float getPrice() { return mPrice; }
67      public final void setPrice(float price) { mPrice = price; }
68      public boolean getHasPicture() { return mHasPicture; }
69      public final void setHasPicture(boolean hasPicture) { mHasPicture = hasPicture; }
70      public Bitmap getPicture() { return mPicture; }
71      public final void setPicture(Bitmap image) {
72          mPicture = image;
73          mHasPicture = (image != null);
74      }
75      public Account getAccount() { return mAccount; }
76      // Setting the Account will automatically set the database foreign key, too.
77      public final void setAccount(Account account) {
78          mAccount = account;
79          mUserId = account.getId();
80      }
81      public int getNumComments() { return mNumComments; }
82      public final void setNumComments(int numComments) {
83          mNumComments = numComments;
84      }
85
86      @Override
87      public boolean equals(Object o) {
88          return o instanceof SaleItem && ((SaleItem)o).mId == mId;
89      }
90  }
91
```

```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
```

```
64      public String getId() { return mId; }
65      public final void setId(String id) { mId = id; }
66      public String getZip() { return mZip; }
67      public final void setZip(String zip) { mZip = zip; }
68      public String getZipType() { return mZipType; }
69      public final void setZipType(String zipType) { mZipType = zipType; }
70      public String getCity() { return mCity; }
71      public final void setCity(String city) { mCity = city; }
72      public String getState() { return mState; }
73      public final void setState(String state) { mState = state; }
74      public String getLocationType() { return mLocationType; }
75      public final void setLocationType(String locationType) { mLocationType = locationType;
    }
76      public String getLocationText() { return mLocationText; }
77      public final void setLocationText(String locationText) { mLocationText = locationText;
    }
78      public double getLat() { return mLat; }
79      public final void setLat(double lat) { mLat = lat; }
80      public double getLong() { return mLong; }
81      public final void setLong(double lng) { mLong = lng; }
82      public String getStateText() { return mStateText; }
83      public final void setStateText(String stateText) { mStateText = stateText; }
84
85      @Override
86      public boolean equals(Object o) {
87          return o instanceof ZipCodes && ((ZipCodes)o).mId == mId;
88      }
89  }
90
```

```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;
    }
}

```

```
/* 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; }

```

```
64      @Override
65      public boolean equals(Object o) {
66          return o instanceof SchoolDistance && ((SchoolDistance)o).mId == mId;
67      }
68  }
69
```

```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();
    }

```

```
64      public static SaleItem getItem(int position) {
65          if (mSaleItems.size() > position) return mSaleItems.get(position);
66          else return null;
67      }
68
69      public static void notifyItem(SaleItem item) {
70          if (mSaleItems.contains(item)) {
71              int position = mSaleItems.indexOf(item);
72              for (RecyclerView.Adapter adapter : mAdapters) {
73                  adapter.notifyItemChanged(position);
74              }
75          }
76      }
77
78      public static void addItem(SaleItem item) {
79          mSaleItems.add(item);
80          for (RecyclerView.Adapter adapter : mAdapters) {
81              adapter.notifyDataSetChanged();
82          }
83      }
84
85      public static void removeItem(int position) {
86          mSaleItems.remove(position);
87          for (RecyclerView.Adapter adapter : mAdapters) {
88              adapter.notifyDataSetChanged();
89          }
90      }
91
92      /*
93         The Refresh executes a new search. It can be called from anywhere.
94         For example, when you change your school, we have to refresh.
95      */
96      private static MobileServiceTable<SchoolDistance> mSchoolDistanceTable;
97      private static MobileServiceTable<Schools> mSchoolsTable;
98      private static MobileServiceTable<Account> mAccountTable;
99      private static MobileServiceTable<SaleItem> mSaleItemTable;
100     private static MobileServiceTable<ItemComment> mItemCommentTable;
101     public static void Refresh(Context context, FblaAzure azure) {
102         if (!azure.getLoggedOn()) return;
103         mSaleItems.clear();
104
105         final int searchMiles = azure.getSearchMiles(context);
106         Account myAccount = azure.getAccount();
107         if (myAccount.getSchool() == null) {
108             for (RecyclerView.Adapter adapter : mAdapters) {
109                 adapter.notifyDataSetChanged();
110             }
111             return;
112         }
113         final String searchUserId = azure.getUserId();
114         final Schools searchSchool = myAccount.getSchool();
115         Log.d("LocalController:Refresh", searchSchool.getId()+" "+searchMiles);
116
117         mSchoolDistanceTable = azure.getClient().getTable(SchoolDistance.class);
118         mSchoolsTable = azure.getClient().getTable(Schools.class);
119         mAccountTable = azure.getClient().getTable(Account.class);
120         mSaleItemTable = azure.getClient().getTable(SaleItem.class);
121         mItemCommentTable = azure.getClient().getTable(ItemComment.class);
122         new AsyncTask<Object, Object, Object>() {
123             @Override
124             protected Object doInBackground(Object... params) {
125                 try {
126                     ArrayList<SaleItem> saleItems = new ArrayList<>();
```

```
127                     // First get all of the schools nearby
128                     Log.d("LocalController:Refresh", "Starting");
129                     final MobileServiceList<SchoolDistance> distances =
130                             mSchoolDistanceTable.where().field("fromid").eq(searchSchool.
    getId())
131                                     .and().field("miles").le(searchMiles)
132                                     .select("id", "fromid", "toid", "miles")
133                                     .execute().get();
134                     for (SchoolDistance toSchool : distances) {
135                         // Get each school details
136                         final Schools school = mSchoolsTable.lookUp(toSchool.getToId()).get
    ();
137                         // Get all accounts for each school
138                         final MobileServiceList<Account> accounts =
139                                 mAccountTable.where().field("schoolid").eq(school.getId()).
    execute().get();
140                         for (Account account : accounts) {
141                             // Now get all the items for each account (excluding your own)
142                             if (!account.getId().equals(searchUserId)) {
143                                 account.setSchool(school);
144                                 final MobileServiceList<SaleItem> items =
145                                         mSaleItemTable.where().field("userid").eq(account.
    getId()).execute().get();
146                                 for (SaleItem item : items) {
147                                     item.setAccount(account);
148                                     // Get its picture
149                                     if (item.getHasPicture())
150                                         item.setPicture(FblaPicture.DownloadImage(item.
    getId()));
151                                     // Finally, count the number of comments that are on
    each item.
152                                     final MobileServiceList<ItemComment> cnt =
153                                             mItemCommentTable.where().field("itemid").eq(
    item.getId()).includeInlineCount().execute().get();
154                                     item.setNumComments(cnt.getTotalCount());
155                                     saleItems.add(item);
156                                 }
157                             }
158                         }
159                     }
160                     return saleItems;
161                 } catch (Exception exception) {
162                     Log.e("LocalController:Refresh", exception.toString());
163                 }
164                 return null;
165             }
166             @Override
167             protected void onPostExecute(Object result) {
168                 Log.d("LocalController:Refresh", "Complete");
169                 if (result != null) {
170                     ArrayList<SaleItem> saleItems = (ArrayList<SaleItem>)result;
171                     for (SaleItem item : saleItems) {
172                         mSaleItems.add(item);
173                     }
174                     for (RecyclerView.Adapter adapter : mAdapters) {
175                         adapter.notifyDataSetChanged();
176                     }
177                     for (RefreshResultListener l : mListeners) {
178                         l.onRefreshComplete();
179                     }
180                 }
181             }
182         }.execute();
```

```
183     }
184
185     private static ArrayList<RefreshResultListener> mListeners = new ArrayList<>();
186     // Add a listener to call after refresh is complete
187     public static void attachRefreshListener(RefreshResultListener listener) {
188         mListeners.add(listener);
189     }
190     public static void detachRefreshListener(RefreshResultListener listener) {
191         mListeners.remove(listener);
192     }
193
194     // This is the interface to use on the logon callbacks.
195     public interface RefreshResultListener {
196         void onRefreshComplete();
197     }
198 }
199
```

```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();
```

```
64          }
65      }
66
67      public static void removeItem(int position) {
68          mSaleItems.remove(position);
69          for (RecyclerView.Adapter adapter : mAdapters) {
70              adapter.notifyDataSetChanged();
71          }
72      }
73
74      private static MobileServiceTable<SaleItem> mSaleItemTable;
75      private static MobileServiceTable<ItemComment> mItemCommentTable;
76      public static void Refresh(FblaAzure azure) {
77          Log.d("MySalesController", "Refresh");
78          if (!azure.getLoggedOn()) return;
79          mSaleItems.clear();
80
81          mSaleItemTable = azure.getClient().getTable(SaleItem.class);
82          mItemCommentTable = azure.getClient().getTable(ItemComment.class);
83          new AsyncTask<Object, Object, Object>() {
84              class TaskResult {
85                  public FblaAzure azure;
86                  public ArrayList<SaleItem> saleItems;
87
88                  public TaskResult (FblaAzure a) {
89                      azure = a;
90                      saleItems = new ArrayList<>();
91                  }
92              }
93
94              @Override
95              protected Object doInBackground(Object... params) {
96                  try {
97                      FblaAzure azure = (FblaAzure)params[0];
98                      TaskResult taskResult = new TaskResult(azure);
99                      final MobileServiceList<SaleItem> result =
100                             mSaleItemTable.where().field("userid").eq(azure.getUserId()).
    execute().get();
101                     for (SaleItem s : result) {
102                         final MobileServiceList<ItemComment> cnt =
103                                 mItemCommentTable.where().field("itemid").eq(s.getId()).
    includeInlineCount().execute().get();
104                         s.setNumComments(cnt.getTotalCount());
105                         // Now get the picture, if it exists.
106                         if (s.getHasPicture())
107                             s.setPicture(FblaPicture.DownloadImage(s.getId()));
108                         taskResult.saleItems.add(s);
109                     }
110                     return taskResult;
111                 } catch (Exception exception) {
112                     Log.e("MySalesController", exception.toString());
113                     return null;
114                 }
115             }
116             @Override
117             protected void onPostExecute(Object result) {
118                 if (result != null) {
119                     TaskResult taskResult = (TaskResult)result;
120                     for (SaleItem item : taskResult.saleItems) {
121                         item.setAccount(taskResult.azure.getAccount());
122                         mSaleItems.add(item);
123                     }
124                     Log.d("MySalesController", "Set Notify");
```

```
125                     for (RecyclerView.Adapter adapter : mAdapters) {
126                         adapter.notifyDataSetChanged();
127                     }
128                 }
129             }
130         }.execute(azure);
131     }
132 }
133
```

```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();
        }
```

```
64          // Update the count on the display, if shown
65          MySalesController.notifyItem(mItem);
66          LocalController.notifyItem(mItem);
67      }
68
69      public static SaleItem getItem() { return mItem; }
70      public static void setItem(SaleItem item) { mItem = item; }
71
72      // Refresh all comments and notify the adapter of the change
73      public static void Refresh(FblaAzure azure) {
74          if (!azure.getLoggedOn()) return;
75          mComments.clear();
76
77          mItemCommentTable = azure.getClient().getTable(ItemComment.class);
78          final MobileServiceTable<Account> mAccountTable = azure.getClient().getTable(
    Account.class);
79          new AsyncTask<Object, Object, Object>() {
80              @Override
81              protected Object doInBackground(Object... params) {
82                  try {
83                      ArrayList<ItemComment> comments = new ArrayList<>();
84                      final MobileServiceList<ItemComment> result =
85                              mItemCommentTable.where().field("itemid").eq(mItem.getId()).
    execute().get();
86                      for (ItemComment comment : result) {
87                          Account account = mAccountTable.lookUp(comment.getUserId()).get();
88                          comment.setAccount(account);
89                          comments.add(comment);
90                      }
91                      return comments;
92                  } catch (Exception exception) {
93                      Log.e("CommentListController", exception.toString());
94                  }
95                  return null;
96              }
97              @Override
98              protected void onPostExecute(Object result) {
99                  // If there are results, copy them into the array and notify the adapter.
100                 // This must be done on the UI thread.
101                 if (result != null) {
102                     ArrayList<ItemComment> comments = (ArrayList<ItemComment>)result;
103                     for (ItemComment comment : comments) {
104                         mComments.add(comment);
105                     }
106                     mItem.setNumComments(comments.size());
107                     for (RecyclerView.Adapter adapter : mAdapters) {
108                         adapter.notifyDataSetChanged();
109                     }
110                 }
111             }
112         }.execute();
113     }
114 }
115
```