Final Write-Up
Michael Grisafe, Lily Samimi and Fei Zhao
Dec 10th, 2014

What is FamCam all about?

Our project would connect geographically distant relatives with their children, whether that is their grandchildren, nieces, nephews, or their own children. Children, especially under the age of 5, love seeing photos of their loved ones and themselves. They are always grabbing their parents smartphones and tablets to look at recent photos. They will look at those pictures on repeat. It's almost impossible to take the phone/tablet away from them. Our application would provide a portal for children to see recent pictures that have been sent from close relatives. This would allow children to stay connected with their distant relatives.

Note: This project is an extension of a project concept created by Christina Pellerano for SI 582 Winter 2014.

Who are the user groups?

For the purpose of this project, we'll be focusing our application on children up to the age of 5 and grandparents.

Why is it important?

Many grandparents are geographically distant from their grandchildren in the United States, restricting their opportunities for interacting with them. Our application places grandchildren and grandparents at the forefront, giving both groups a simple interface in which they can use to upload and view photos at any time in a safe, simple, and secure environment..

Who are the competitors?

Facebook: Grandparents can sign up Facebook pages and parents can post their child's photos on Facebook pages for grandparents.
Instagram: Parents can post their child's pictures on Instagram and share with grandparents.
Skype: Grandparents can participate in video chats with their grandchildren via Skype.
Children entertainment Apps: Apps like "Elmo Calls," which simulates the popular Sesame Street puppet speaking to children and other child-centered game apps.

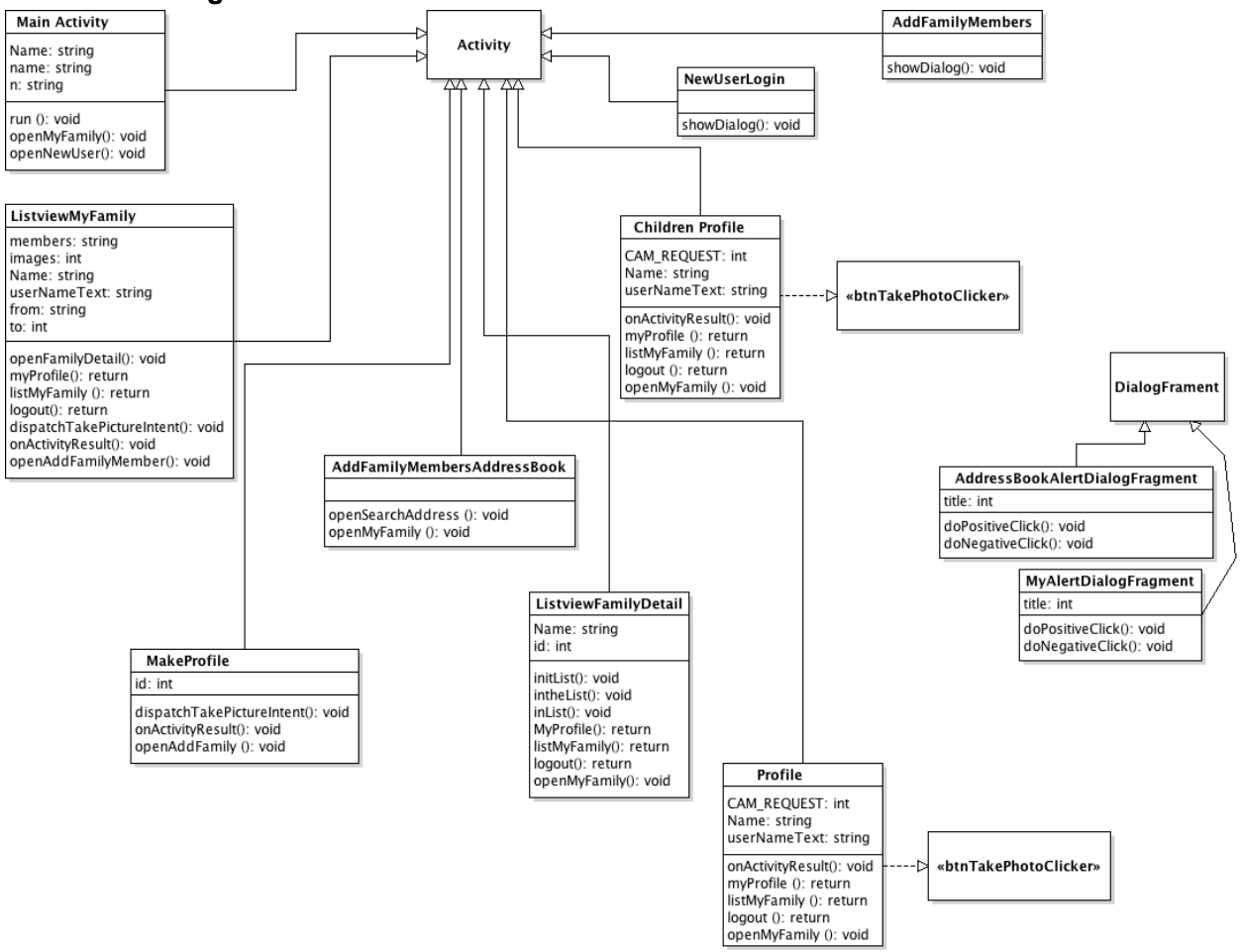
Why is our idea better?

This application is more secure and private than applications like Facebook and Instagram because families would be restricted to connecting to each other. In addition, our application is the only child-centric program on its kind that provides a simplified interface that

children can easily use to look at pictures and take pictures for their grandparents.
By contrast, Instagram and Facebook are especially complicated for young children and older adults, leaving opportunities for accidental posts and security issues which are not present in our proposed app. Other child apps only focus on gaming rather than communication.

NOTE: Our use case for adults can be accessed by inputting the name “Grandma Tata” in the username field on the landing screen. Any other name will simulate the use case for a child. Whether a user is classified as an adult or a child would be handled on the backend of our app after the user inputs their age in the new user registration screen. Users over the age of 10 will be able to have full access to the system and be classified as “adults” for the purpose of this app (functionally, this simply means that they have access to adding other family members).

UML Class Diagram



Michael's Fragment Functionality

The functionality which I am going to write about is alert dialog fragments. We utilized alert dialog fragments throughout our application as a way of warning users when our application was going to access information on the user's phone which may be considered private (e.g. the user's phone list of contacts) or as a final way of asking the user if the information they had created was correct (new profile creation. All of our alert dialogs use similar code, so I will be looking at the alert dialog fragment method within the "NewUserLogin.java" in which new users can submit their registration information (Figure 1.).

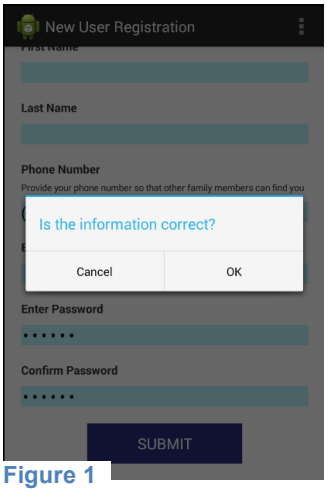


Figure 1

1. Within the onCreate method, I instantiated an object of class Button named "button" which is connected to the XML of ID= "submit" (Figure 2). This ID is located using findViewById. After this button object is created, I evoke the "setOnClickListner" on it, with a method inside it which directs the program via the onClick method to go to the "showDialog()" method when the submit button is clicked.

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


    Button button = (Button) findViewById(R.id.submit);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showDialog();
        }
    });
```

Figure 2

2. Once the submit button is clicked, the showDialog() method is called. Within this method an object of class DialogFragment named "newFragment" is instantiated and set to "MyAlertDialogFragment" (Figure 3). The .newInstance() code sets the text that will appear above the dialog choices ("Is the information correct?"). The dialog fragment itself is called by looking up the item "dialog" from the "Fragment Manager."

```
void showDialog() {
    DialogFragment newFragment = MyAlertDialogFragment
            .newInstance("Is the information correct?");
    newFragment.show(getFragmentManager(), "dialog");
}
```

Figure 3

3. The "MyAlertDialogFragment" referenced above is a subclass of "DialogFragment," a native android class (Figure 4). After extending DialogFragment to MyAlertDialogFragment the fragment is instantiated and the variable "title" string ("Is this information correct?") is "put" into title key.

```
public static class MyAlertDialogFragment extends DialogFragment{

    public static MyAlertDialogFragment newInstance(int title){
        MyAlertDialogFragment frag = new MyAlertDialogFragment();
        Bundle args = new Bundle();
        args.putInt("title",title);
        frag.setArguments(args);
        return frag;
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState){
        int title = getArguments().getInt("title");

        return new AlertDialog.Builder(getActivity())
                .setTitle(title)
                .setPositiveButton("OK",
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog,
                                    int whichButton) {
                                ((NewUserLogin)getActivity())
                                        .doPositiveClick();
                            }
                        })
                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                ((NewUserLogin) getActivity())
                                        .doNegativeClick();
                            }
                        }).create();
    }
```

Figure 4

4. In the next section of code, the AlertDialog builder is called and a new alert object is instantiated (Figure 4). “Ok” is set to the “positive button” via “setPositiveButton” code. OnClickListener is evoked so that the program can tell if the button is clicked, and then the method “.doPositiveClick() is evoked if it is clicked. Similar code is used for the “Cancel” button, directing the program to “.doNegativeClick” method. “.create()” creates the alert fragment.

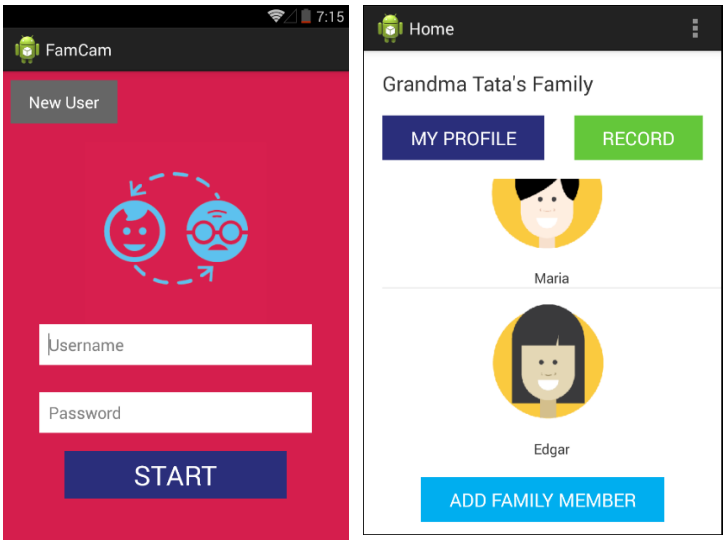
```
public void doPositiveClick() {
    Intent intent = new Intent(this, MakeProfile.class);
    startActivity(intent);
}

public void doNegativeClick() {
```

Figure 5

5. Lastly are the methods directing what the program should do if a positive (“Ok”) or negative (“Cancel”) button are clicked. The doPositiveClick creates a new object of class intent that opens the MakeProfile.class and its corresponding activity. Because we wanted the user to stay on the activity if “Cancel” is clicked the method for doNegativeClick is left empty

Lily’s Shared Preferences Functionality

The functionality that I implemented that I am most proud of is the shared preferences functionality for start screen and subsequent My Family screen. We used shared preferences here to transfer over the user’s name to the My Family screen as well as allow for certain settings to appear based on specific users. For example, in our beta version, if the user types in Grandma Tata then “Grandma Tata’s Family” will appear on the My Family page and an “Add Family Member” button will appear at the bottom of that page.



MainActivity.java

The local variables here that I define are TextView name and String Name = “nameKey”.

I then created the Shared Preferences objects.

Then I call the local variable name and set that equal to a method findViewById which will find the text the user puts into the username box called "editTextName”.

Then using the Shared Preferences object, I use the method for Shared Preferences to save the text that was entered in the username box.

```
public class MainActivity extends Activity {

    TextView name;
    public static final String MyPREFERENCES ="MyPrefs";
    public static final String Name = "nameKey";

    SharedPreferences sharedpreferences;

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


     //code for save the name
        name = (EditText)findViewById(R.id.editTextName);

        //We should always use getDefaultSharedPreferences unless we want to separat
        sharedpreferences = PreferenceManager.getDefaultSharedPreferences(this);

        if (sharedpreferences.contains(Name)){
            name.setText(sharedpreferences.getString(Name,""));
        }

    }
// when run is called the name in edit text view is put to shared preferences
    public void run(View view){
        String n = name.getText().toString();
        Editor editor = sharedpreferences.edit();
        editor.putString(Name,n);

        editor.apply();
    }
```

I added a conditional here to set the text in the username box for the name that was saved in shared preferences.

I then created a method called run and variable string n to get the text that is entered in the text view and put the string in a key, value pair in shared preferences.

ListViewMyFamily.java

I first call the Shared preferences object and use the method to get the saved name.

I then created a variable string usernameText and set that equal to the Shared preferences object using the getString method in a key value pair.

```
public static final String Name = "nameKey";

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

    SharedPreferences mySharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
    String usernameText = mySharedPreferences.getString(Name, "");
    Button button_add = (Button) findViewById(R.id.button_add);
        if (usernameText.equals("Grandma Tata")) button_add.setVisibility(View.VISIBLE);
       else button_add.setVisibility(View.GONE);
```

```
TextView textView= (TextView) findViewById(R.id.replaceName);
textView.setText(usernameText+ "'s Family");
```

I then called the method button and told it to find the button that is the xml.

There is a conditional here if the username text variable is equal to “Grandma Tata” then using a method set the button to visible else set the button to invisible.

Using the textView textView object I then use the findViewbyId method to find the text view that is in the xml and replace the text with the usernameText variable.

LIST VIEW

FamCam Project

Fei Zhao

12/18/2014

Set up the list view in xml file

For every item in list view, we have name and picture. So, we create a new layout resource file with an image view and a text view.

In ListViewMyFamily java class:

Create an array of names and an array of the image id.

Create a hasp map to combine names and images and put them in an array list.

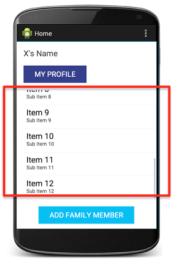
Create a new simple adapter to map the array list to the views in xml file.

Use the adapter to list view the items.

Make every item clickable by using onClick listener, when click on the items, they will do the openFamilyDetail method.

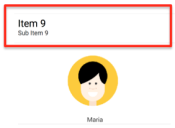
In activity_listview_myfamily.xml

```
<ListView
    android:id="@+id/listView"
    android:layout_width="fill_parent"
    android:layout_height="300dp"
    android:textAlignment="gravity"
    android:smoothScrollbar="true"
    android:stackFromBottom="true"
    android:background="#ffffff"
    android:layout_gravity="center"
    android:minHeight="800dp"
    android:paddingLeft="20dp">
</ListView>
```

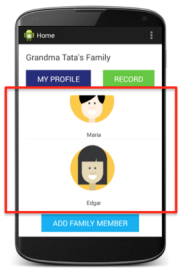


In singlerow.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <ImageView
        android:id="@+id/image"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:contentDescription=""
        android:paddingTop="10dp"
        android:paddingRight="10dp"
        android:layout_gravity="center"
        android:paddingBottom="10dp"/>
    <TextView
        android:id="@+id/txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textSize="15dp"/>

</LinearLayout>
```



In ListviewMyfamily.java




```
String[] members = new String[]{
        "Grandma Tata",
        "Grandpa John",
        "Maria",
        "Edgar"
};

int[] images = new int[]{
        R.drawable.man,
        R.drawable.woman,
        R.drawable.man2,
        R.drawable.woman2
};
List<HashMap<String,String>> aList = new ArrayList<HashMap<String, String>>();
for (int i=0;i<4; i++) {
    HashMap<String,String> hm = new HashMap<String, String>();
    hm.put("txt","" + members[i]);
    hm.put("image",Integer.toString(images[i]));
    aList.add(hm);
}
```

```
String[] from = {"image","txt"};

int[] to = {R.id.image,R.id.txt};

SimpleAdapter adapter = new SimpleAdapter(getBaseContext(),aList,R.layout.single_row,from,to);

ListView listView = (ListView) findViewById(R.id.listView);

listView.setAdapter(adapter);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    public void onItemClick(AdapterView<?> parentAdapter, View view, int position,
                            long id) {
        openFamilyDetail(id);

    }
});
```

LIST VIEW

FamCam Project

Fei Zhao

12/18/2014

Send id by EXTRA_MESSAGE to FamilyDetail class.

Define the openFamilyDetail method.

Setup text view and image view in family detail xml file.

Create array list of family member image, name and description. Define three inlist methods.

Use the id message from my family activity to show the image and text view content.

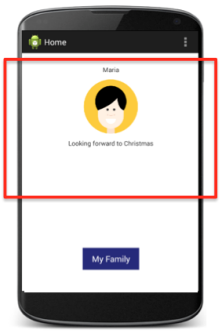
So, when you click different family member, their information will show in the detail page by the id of that member.

In ListviewMyfamily.java

```
public final static String EXTRA_MESSAGE = "com.lilysamimi.famcamproject.MESSAGE";

public void openFamilyDetail(long id) {
    Intent intent = new Intent(this, ListviewFamilyDetail.class);
    String message = String.valueOf(id);
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

In activity_listview_familydetail.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txt"
    android:layout_gravity="center"
    />
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:id="@+id/img"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/des"
    android:layout_gravity="center"
    />
```



In ListviewFamilyDetail.java

```
// defined the initList method, add these names into the ArrayList called memberList
private void initList() {
    membersList.add("Grandma Tata");
    membersList.add("Grandpa John");
    membersList.add("Maria");
    membersList.add("Edgar");
}

//created a new ArrayList of strings, named memberList, it will list all the family member name
List<String> membersList = new ArrayList<String>();


// defined the intheList method, add these descriptions into the ArrayList called description
private void intheList() {
    description.add("Love cooking for my family :D");
    description.add("I want to go fishing!");
    description.add("Looking forward to Christmas");
    description.add("Watching football game");
}

//created a new ArrayList of strings, named description, it will list all the family member's description

List<String> description = new ArrayList<>();

private void inList() {

    array_image.add(R.drawable.man);
    array_image.add(R.drawable.woman);
    array_image.add(R.drawable.man2);
    array_image.add(R.drawable.woman2);
}
//created a new ArrayList of integer

ArrayList<Integer> array_image = new ArrayList<Integer>();

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


    initList();
    inList();
    intheList();


    Intent intent = getIntent();
    String message = intent.getStringExtra(ListviewMyFamily.EXTRA_MESSAGE);

    int id = (int) Long.parseLong(message);

    //create the text view

    TextView textView=(TextView)findViewById(R.id.txt);

    textView.setText(membersList.get(id));

    TextView descriptions=(TextView)findViewById(R.id.des);

    descriptions.setText(description.get(id));

    //create the image view

    ImageView imageView=(ImageView)findViewById(R.id.img);

    imageView.setImageResource(array_image.get(id));

}
```