CS 442: Mobile Applications Development

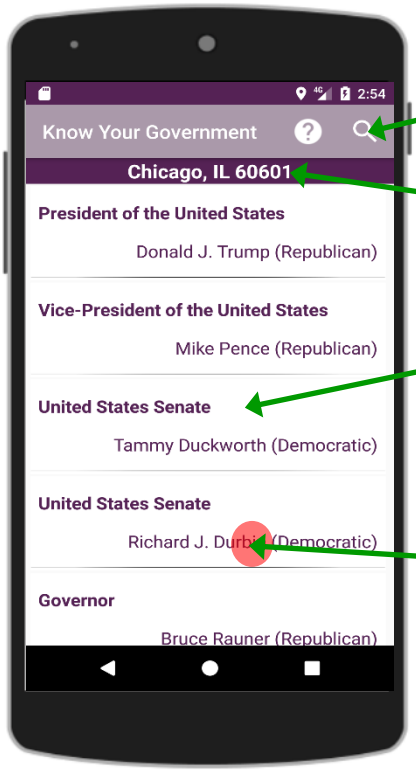
Assignment 4 – Know Your Government *(400 pts)*

Uses: Location Services, Internet, Google APIs, Images, Picasso Library, Implicit Intents, TextView Links

App Highlights:

- This app will acquire and display an interactive list of political officials that represent the current location (or a specified location) at each level of government.
- Android location services will be used to determine the user's location.
- The Google Civic Information API will be used to acquire the government official data (via REST service and JSON results).
- You *will* need to use a different layout for landscape orientation for 2 of the activities in this application. Those details are specified later in the document.
- Clicking on an official's list entry opens a detailed view of that individual government representative.
- Clicking on the photo of an official will display a Photo Activity, showing a larger version of the photo.
- An "About" activity will show application information (Author, Copyright data & Version)
- Your manifest should add permissions for ACCESS_FINE_LOCATION and INTERNET
- The application is made up of 4 activities, shown below:

1) Main Activity



Options Menu items for "about" information and manual location entry

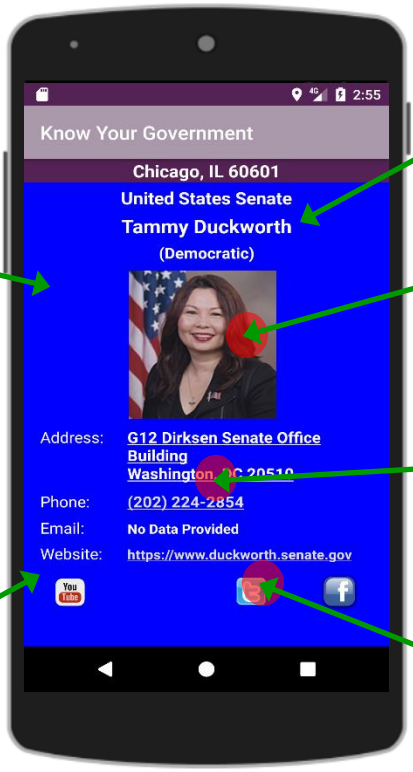
Display of current location (or user-specified location)

RecyclerView list of government officials (List scrolls up & down)

Click on a government official entry to open a new activity containing detailed information on the selected individual.

No separate landscape layout is needed for the Main Activity.

2) *Official Activity*



A ScrollView must be used in case the information does not completely fit on the visible screen.

Basic official data
- *Office (i.e., United States Senate)*
- *Name (i.e., John E. Smith)*
- *Party (i.e., Democratic)*

Photo of official (where available). NOTE, a default image should be displayed when a photo is not specified.

Clicking photo opens Photo Detail Activity

Contact Information (as available)
(all are clickable – implicit intents)
- *Office Address*
- *Phone Number*
- *Email address*
- *Website*

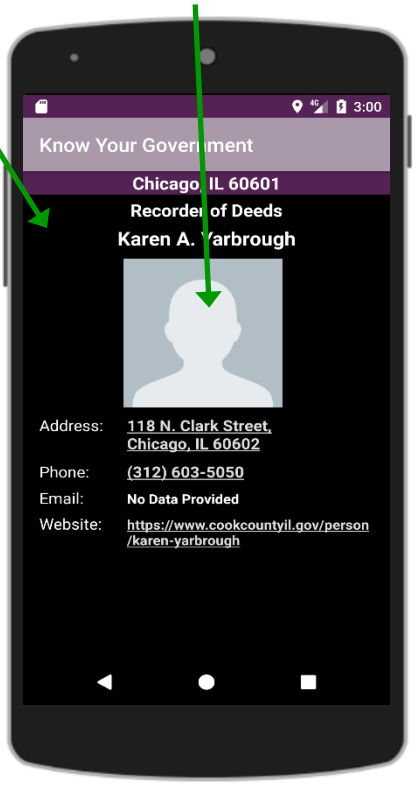
Social Media (as available)
(all are clickable – implicit intents)
- *Facebook*
- *Twitter*
- *Google+*
- *YouTube*

The background color of this activity is based upon the official's political party:
Republican = RED,
Democratic = BLUE,
Otherwise use BLACK

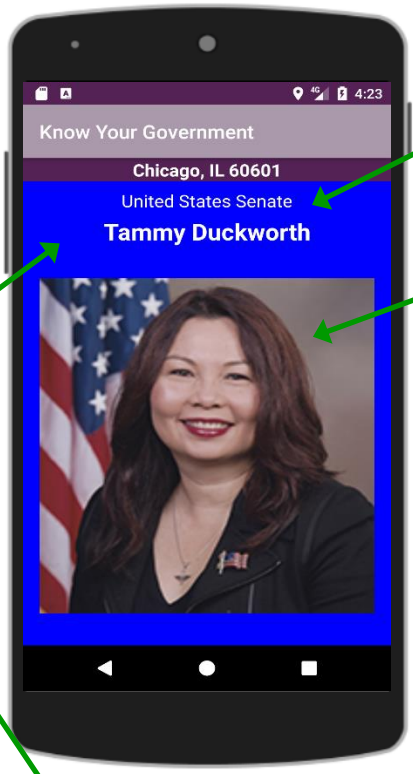
NOTE, a default image should be displayed when a photo is not specified.

Clicking photo opens Photo Detail Activity

Placeholder image displayed while photo is loading



3) Photo Detail Activity



Basic official data
- *Office (i.e., United States Senate)*
- *Name (i.e., John E. Smith)*

Full-sized image of official (where available).

NOTE: This activity should not open if the representative's image is not specified.

The background color of this activity is based upon the official's political party:
Republican = RED,
Democratic = BLUE,
Otherwise use BLACK

Placeholder image displayed while photo is loading

4) About Activity

Application information (Author, Copyright data & Version)

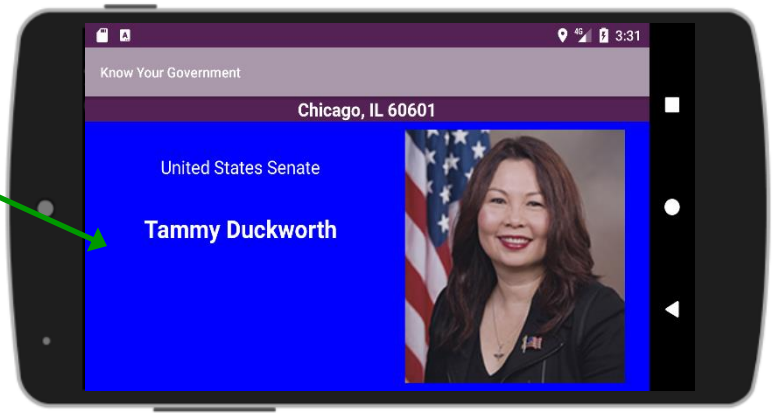
Full-sized background image

No separate landscape layout is needed for the About Activity.

Required Landscape Layouts

Landscape Layout for Official Activity

Landscape Layout for Photo Activity



Internet Data:

Downloading data for the government officials requires a download via the Google Civic Information API. The Civic Information API lets you enter a zip code or city name to look up the corresponding district at each level of government and the names and social media properties for the elected officials in those districts.

NOTE: You MUST sign up Google to get an API key. Your API KEY must be supplied with all Google Civic Information API queries. You can get an API key by following the instructions at:

https://developers.google.com/civic-information/docs/using_api

Follow the instructions for "Acquiring and using an API key". The type of credential you need to create is "API Key" (not "OAuth client ID" and not "Service account key"). This is a very quick and easy process. The API key will be a long string of characters. The results of Civic Information API calls are returned in JSON format. The content of the returned results is described later in this section.

Query Format: https://www.googleapis.com/civicinfo/v2/representatives?key=*Your-API-Key*&address=*zip-code*
https://www.googleapis.com/civicinfo/v2/representatives?key=*Your-API-Key*&address=*city*

For example, if your API Key was "ABC123xyz" and the zip code was 60605, your full URL would be:
https://www.googleapis.com/civicinfo/v2/representatives?key=ABC123xyz**&address=**60605

For example, if your API Key was "ABC123xyz" and the city was Chicago, your full URL would be:
https://www.googleapis.com/civicinfo/v2/representatives?key=ABC123xyz**&address=**Chicago

Google Civic Information API Results Example:

The JSON results you receive contains 4 sections, described in detail below. The *normalizedInput* section contains location details for the results provided. The *divisions* section lists political geographic divisions, like a country, state, county, or legislative district *(we do not need this section for our application).* The *offices* section lists the political positions governing the specified location. The *officials* section lists people presently serving in the offices specified in the *offices* section.

High-level view of JSON results:

```
{
    "kind": "civicinfo#representativeInfoResponse", ← We do not need this
    "normalizedInput": {   ← We need this section, it is accessed as a JSONObject
        … ← Data will be here
    },
    "divisions": {  ← We do not need this
        … ← Data will be here, we do not need this section
    },
    "offices": [   ← We need this section, it is accessed as a JSONArray
        … ← Data will be here
    ],
    "officials": [   ← We need this section, it is accessed as a JSONArray
        … ← Data will be here
    ]
}
```

JSON Section Detail:

The "normalizedInput" JSONObject contains the following:

```
"normalizedInput": {
        "line1": "",
        "city": "Chicago",
        "state": "IL",
        "zip": "60654"
},
```

← We want this for the location display in our activities

← We want this for the location display in our activities

← We want this for the location display in our activities

The "divisions" section comes next, but we do not need this section so it is not described here.

The "offices" JSONArray contains multiple instances of the following:

```
"offices": [
  {
   "name": "President of the United States",
   "divisionId": "ocd-division/country:us",
   "levels": [
    "country"
   ],
   "roles": [
    "headOfState",
    "headOfGovernment"
   ],
   "officialIndices": [
    0
   ]
  },
```

We want this, the "office" title the representative holds

We want this, it is the index into the "officials" JSONArray (see the next section) which contains the details of the person that holds this office.

NOTE: There can be more than one index as time offices have multiple representatives.

```
  {
   "name": "United States Senate",
   "divisionId": "ocd-division/country:us/state:il",
   "levels": [
    "country"
   ],
   "roles": [
    "legislatorUpperBody"
   ],
   "officialIndices": [
    2,
    3
   ]
  },
```

We want this, the "office" title the representative holds

We want this, it is the index into the "officials" JSONArray (see the next section) which contains the details of the person that holds this office.

NOTE: There can be more than one index as time offices have multiple representatives.

```
  …
```

The above sections repeat many times, once per government office

```
],
```

The "officials" JSONArray contains multiple instances of the following:

This is the first official (index 0). This index corresponds to the `"officialIndices"` *we found in the* `"offices"` *section above.*

Note: For any text data on this page that is not supplied, use the default String "No Data Provided"

```
"officials": [
  {
    "name": "Donald J. Trump",
    "address": [
      {
        "line1": "The White House",
        "line2": "1600 Pennsylvania Avenue NW"
        "city": "Washington",
        "state": "DC",
        "zip": "20500"
      }
    ],
    "party": "Republican",
    "phones": [
      "(202) 456-1111"
    ],
    "urls": [
      "http://www.whitehouse.gov/"
    ],
    "emails": [
      "email@address.com"
    ]
    "photoUrl": "https://www.whitehouse.gov/sites/whitehouse.gov/files/images/45/PE%20Color.jpg",
    "channels": [
      {
        "type": "GooglePlus",
        "id": "+whitehouse"
      },
      {
        "type": "Facebook",
        "id": "whitehouse"
      },
      {
        "type": "Twitter",
        "id": "whitehouse"
      },
      {
        "type": "YouTube",
        "id": "whitehouse"
      }
    ]
  },
  {
    … ← The above section repeats many times, once per government official. The second official in this JSONArray is index 1, the third second official in this JSONArray is index 2, and so on.
  }
]
```

This is the name of the person that holds this office.

This is the Address (check for line1, line2 & line3 – concatenate them into one String), City, State & Zip Code of this person's office

This is the political party of this person: "Republican", "Democratic/ Democrat", or "Unknown". Note this section might also be skipped – consider that as party "Unknown".

This person's office phone number. There may be more than one – just use the first entry. Note this section might also be skipped.

This person's office web site. There may be more than one – just use the first entry. Note this section might also be skipped.

This person's office email address. There may be more than one – just use the first entry. Note this section might also be skipped.

This is the URL to the person's photo. Note this section might also be skipped. In this case, use a "place holder" photo.

These are the user ids for the related social media channels. There will be up to four entries. Note this section might also be skipped. Possible entries are:

- *GooglePlus*
- *Facebook*
- *Twitter*
- *YouTube*

Using Picasso for Photo Downloads

The downloading of the photos of the officials in the Official Activity and the Photo Activity must use the Picasso library (as was discussed in class). Note that some photo URLs try to use URL forwarding which is done automatically in web browsers. This is not done automatically in our case. So, if the http phot link fails, we should re-try the same url as https. This can be done as follows:

```
if (<your official's photo url> != null) {

    Picasso picasso = new Picasso.Builder(this).listener(new Picasso.Listener() {

        @Override
        public void onImageLoadFailed(Picasso picasso, Uri uri, Exception exception) {

      // Here we try https if the http image attempt failed
            final String changedUrl = viewOffice.photoUrl.replace("http:", "https:");
            picasso.load(changedUrl)
                    .error(R.drawable.brokenimage)
                    .placeholder(R.drawable.placeholder)
                    .into(officialImageView);
        }
    }).build();

    picasso.load(viewOffice.photoUrl)
            .error(R.drawable.brokenimage)
            .placeholder(R.drawable.placeholder)
            .into(officialImageView);

} else {
    Picasso.get().load(viewOffice.photoUrl)
            .error(R.drawable.brokenimage)
            .placeholder(R.drawable.missing)
            .into(officialImageView);
}
```

Social Media Implicit Intent Examples

The following are examples of how you should code the 4 social-media implicit intents (three of these have been seen already in earlier examples, but are reproduced here for convenience):

Twitter (example onClick method to be associated with the Twitter ImageView icon):

```
public void twitterClicked(View v) {
    Intent intent = null;
    String name = <the official's twitter id from download>;
    try {
        // get the Twitter app if possible
        getPackageManager().getPackageInfo("com.twitter.android", 0);
        intent = new Intent(Intent.ACTION_VIEW, Uri.parse("twitter://user?screen_name=" + name));
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    } catch (Exception e) {
        // no Twitter app, revert to browser
        intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://twitter.com/" + name));
    }
    startActivity(intent);
}
```

Facebook (example onClick method to be associated with the Facebook ImageView icon):

```
public void facebookClicked(View v) {
    String FACEBOOK_URL = "https://www.facebook.com/" + <the official's facebook id from download>;
    String urlToUse;

    PackageManager packageManager = getPackageManager();
    try {
        int versionCode = packageManager.getPackageInfo("com.facebook.katana", 0).versionCode;
        if (versionCode >= 3002850) { //newer versions of fb app
            urlToUse = "fb://facewebmodal/f?href=" + FACEBOOK_URL;
        } else { //older versions of fb app
            urlToUse = "fb://page/" + channels.get("Facebook");
        }
    } catch (PackageManager.NameNotFoundException e) {
        urlToUse = FACEBOOK_URL; //normal web url
    }
    Intent facebookIntent = new Intent(Intent.ACTION_VIEW);
    facebookIntent.setData(Uri.parse(urlToUse));
    startActivity(facebookIntent);
}
```

GooglePlus (example onClick method to be associated with the GooglePlus ImageView icon):

```
public void googlePlusClicked(View v) {
    String name = <the official's google plus id from download>;
    Intent intent = null;
    try {
        intent = new Intent(Intent.ACTION_VIEW);
        intent.setClassName("com.google.android.apps.plus",
                "com.google.android.apps.plus.phone.UrlGatewayActivity");
        intent.putExtra("customAppUri", name);
        startActivity(intent);
    } catch (ActivityNotFoundException e) {
        startActivity(new Intent(Intent.ACTION_VIEW,
                                 Uri.parse("https://plus.google.com/" + name)));
}
```

YouTube (example onClick method to be associated with the YouTube ImageView icon):

```
public void youTubeClicked(View v) {
    String name = <the official's youtube id from download>;
    Intent intent = null;
    try {
        intent = new Intent(Intent.ACTION_VIEW);
        intent.setPackage("com.google.android.youtube");
        intent.setData(Uri.parse("https://www.youtube.com/" + name));
        startActivity(intent);
    } catch (ActivityNotFoundException e) {
        startActivity(new Intent(Intent.ACTION_VIEW,
                                 Uri.parse("https://www.youtube.com/" + name)));
    }
}
```

Provided Icons

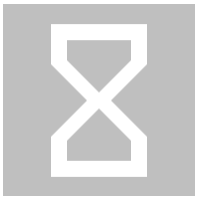
To insure a consistent and professional appearance of our application, the following image files are being provided to you for use in this assignment. You can simply download them, add them to your Android Studio project, and use them with your ImageViews.

Background image for About Activity

Image to use for Official's without an image URL

Image to use for bad photo URLs (the background of this image is transparent)

Image to use for bad photo URLs (the background of this image is transparent)



Separator image to add to the end of your list entry layout

Facebook icon

GooglePlus icon

Twitter icon

YouTube icon

Launcher Icon

Application Behavior Diagrams:

1) App MainActivity

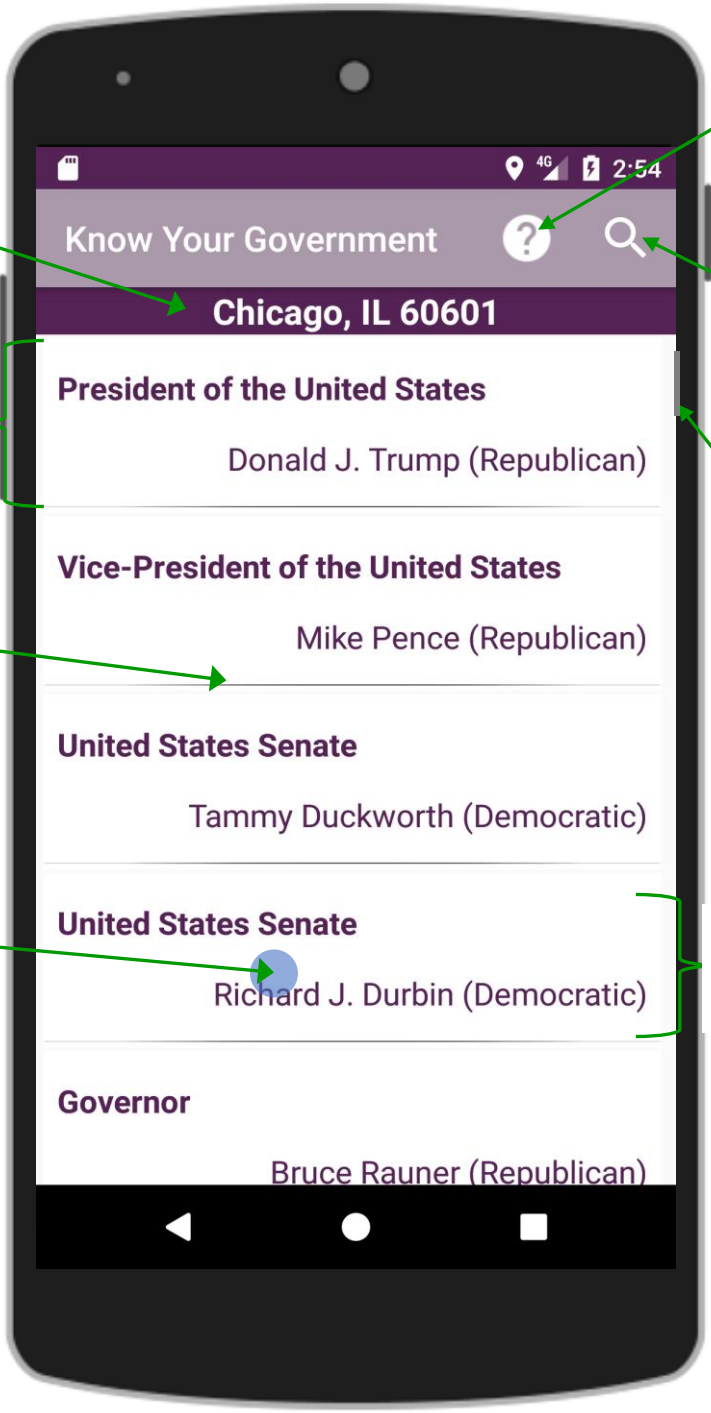
On startup, the application should use the device's current location zip code to populate the list with officials.

Full location information comes from the normalizedInput section of the JSON download.

Each political official entry displays the office (upper-left) and the official's name and political party (lower-right). The political party is in parenthesis.

A separator image should be added to the end of each list entry to create a nice break between entries.

Click on an entry to open the individual Representative Activity



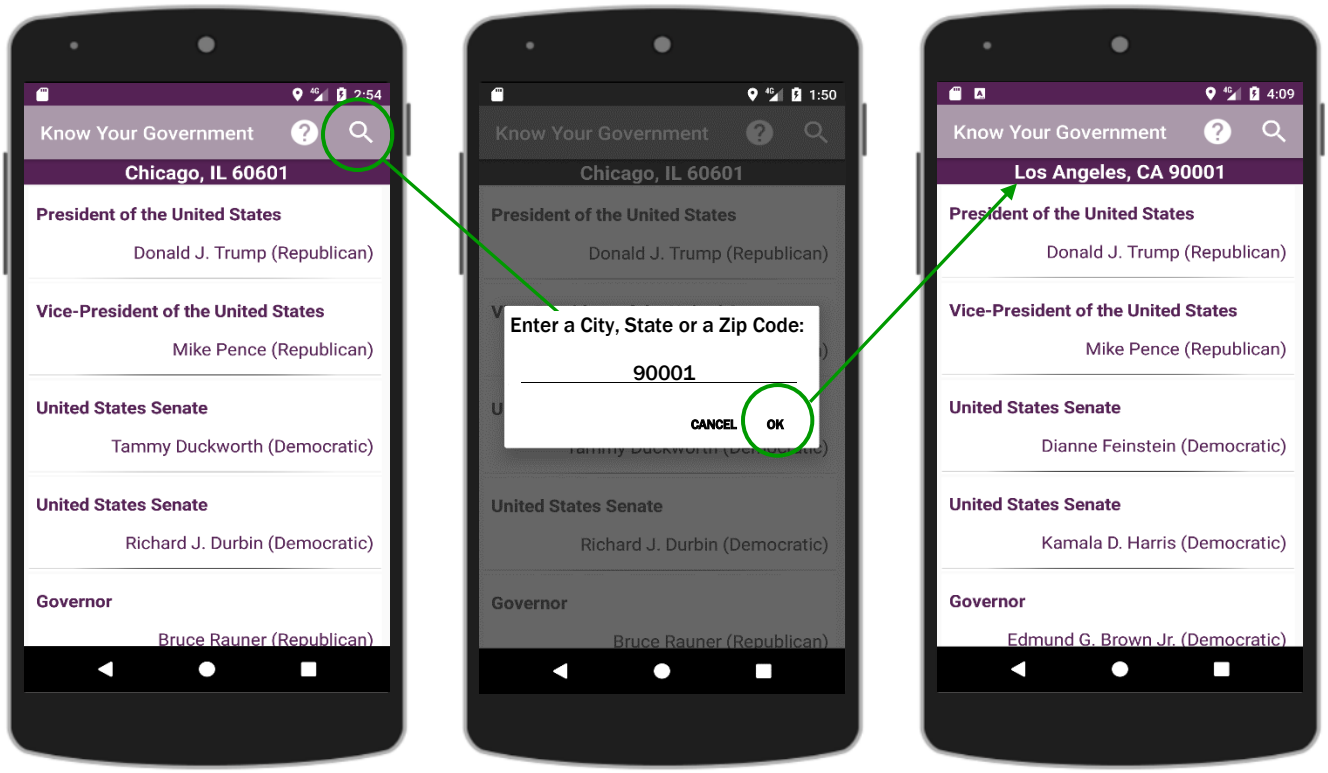
The "info" options menu item opens the "About" activity.

The "Location" options menu item opens an alert dialog that allows the user to enter a city/state or zip code manually.

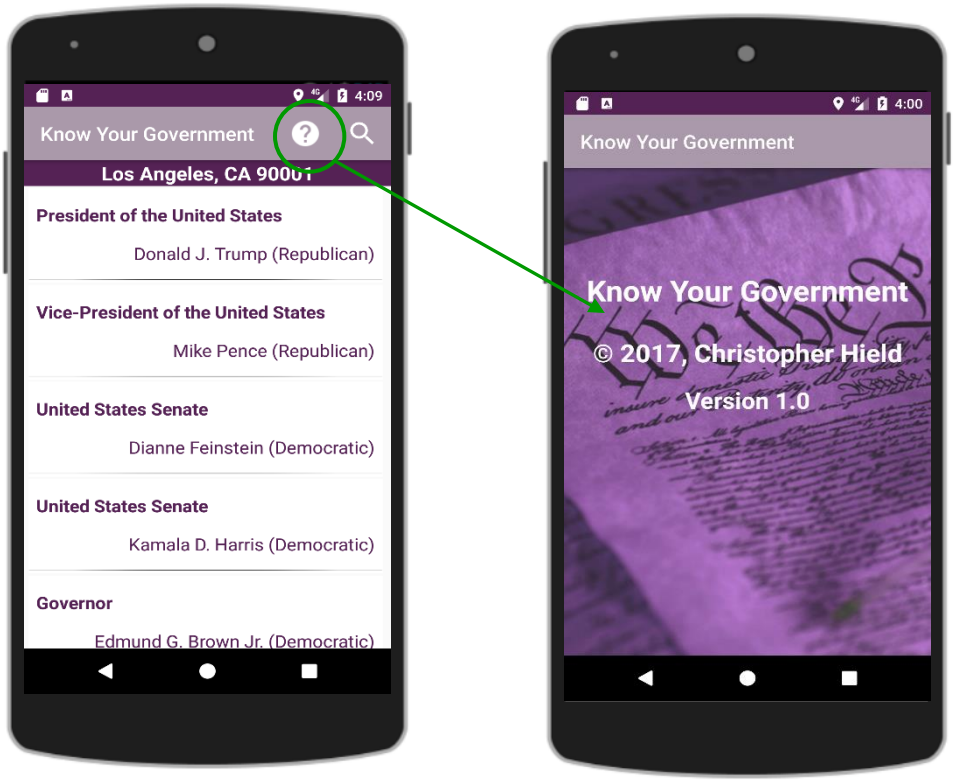
A scrollbar should be present along the right-hand side

RecyclerView list entries have their own layout

2) Manually Setting the location:

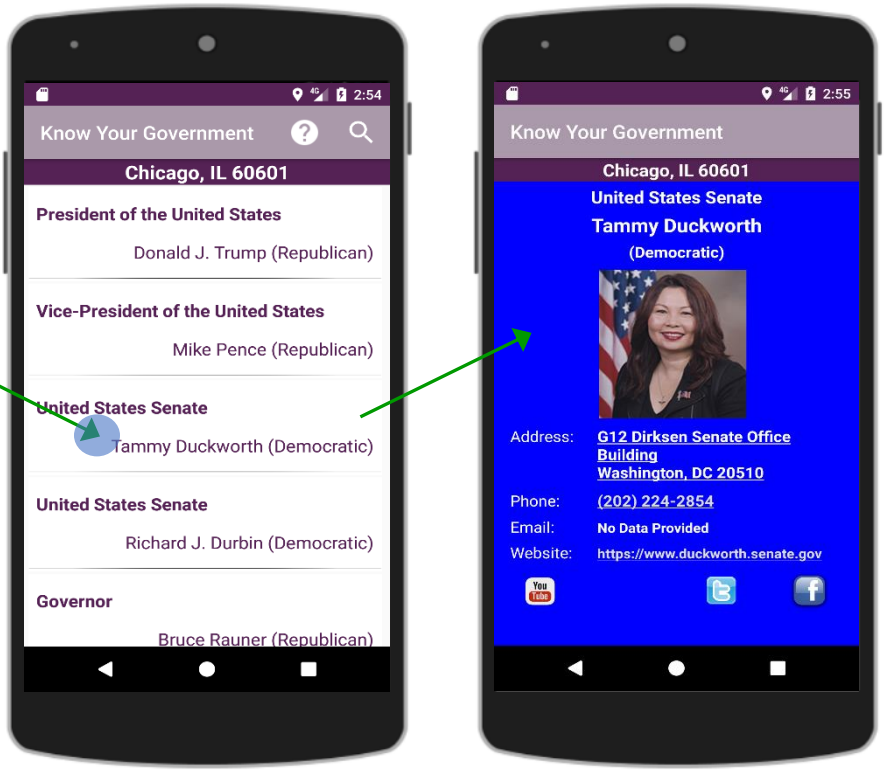


3) Opening the About Activity



4) Viewing an individual Official:

Click on an entry to open the individual Official Activity



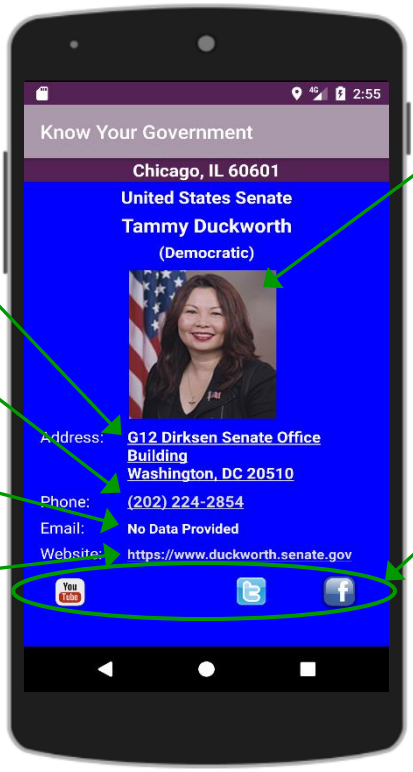
5) Interactive links on the official's activity:

Click on the address (where available) to open the office location in Google Maps

Click on the phone number (where available) to open the Phone App with the phone number loaded

Click on the Email Address (where available) to open the email app with this address pre-loaded in the new message

Click on the Website address (where available) to open the website in a browser.

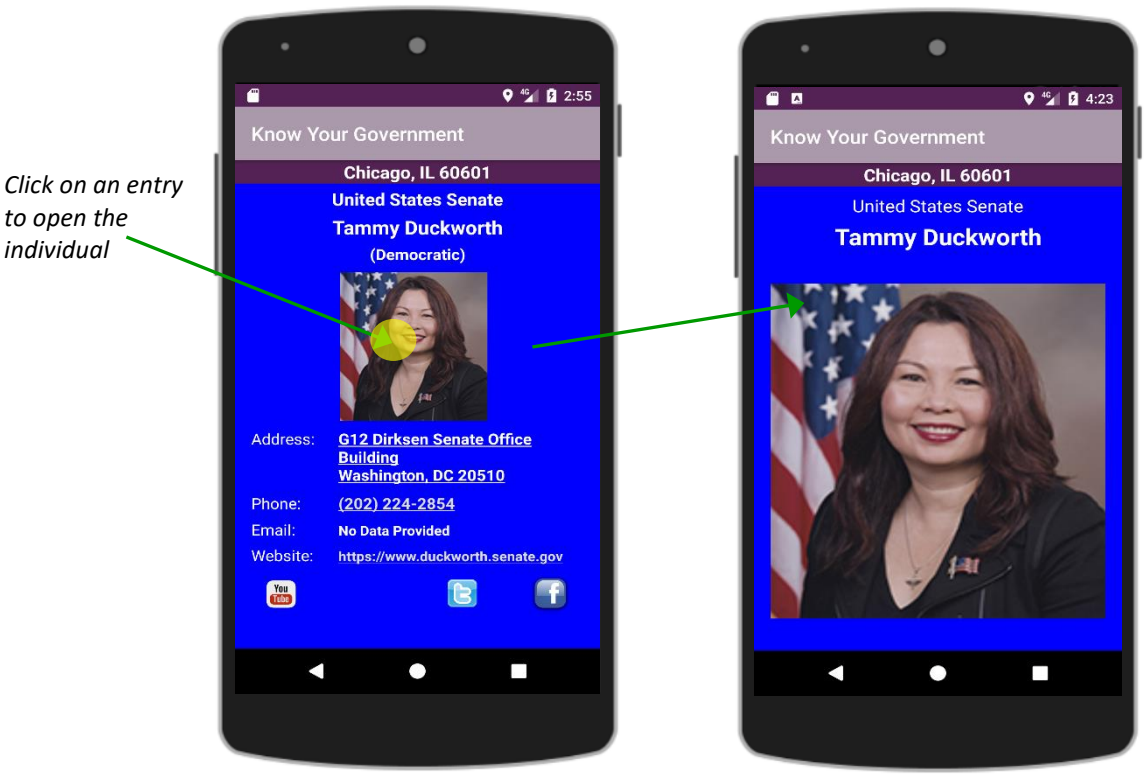


Click on the photo (where available) to open the Photo Activity

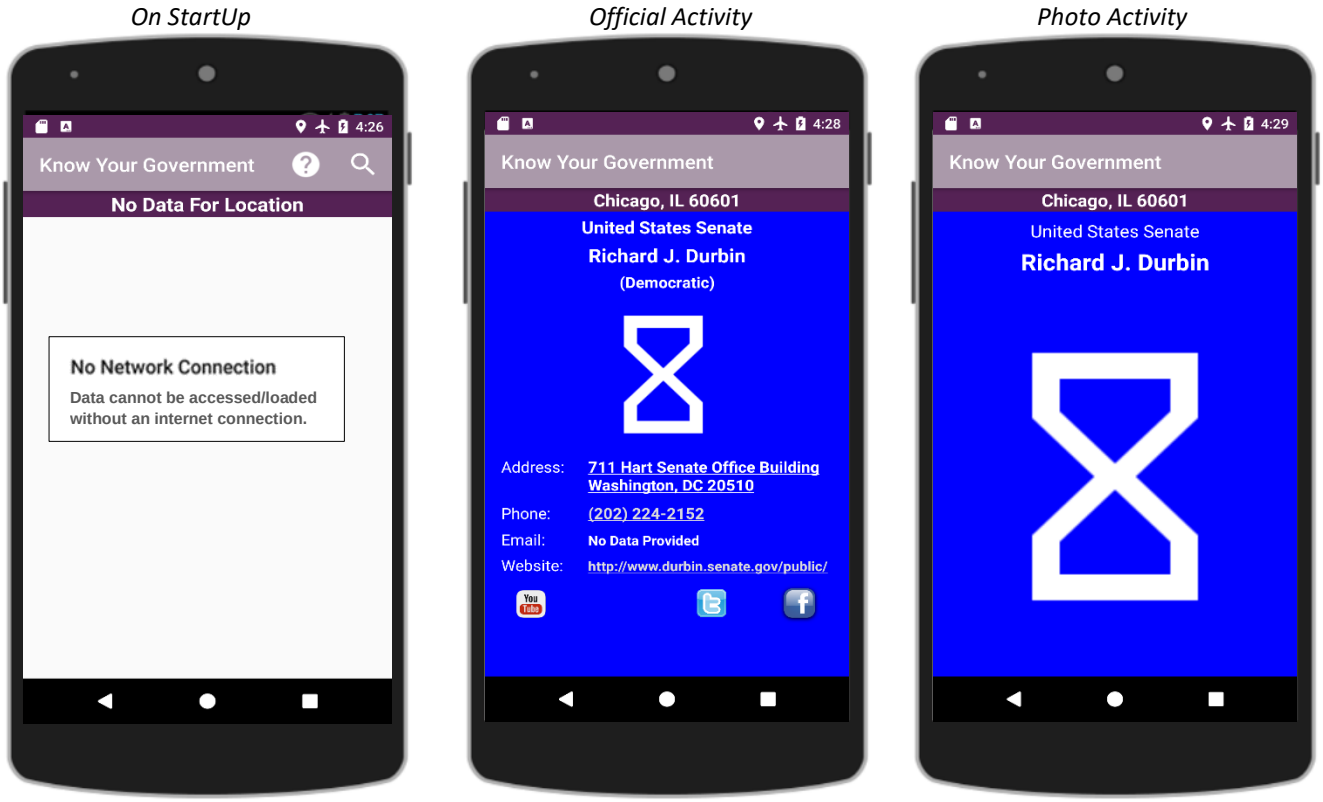
Social media links. Where supplied, the official's social media links will be displayed using the social media's icon. If available, the icon is present - if not it is absent (the icon should not be displayed).

Clicking on these will open the related app (if installed) or will default to opening the site in a web browser.

6) Viewing an individual Official's Photo:



7) If no internet connection:



Development Plan

1) Create the base app:
    a. MainActivity with RecyclerView
    b. Official Class
    c. RecyclerView Adapter
    d. RecyclerView ViewHolder
    e. Create fake "dummy" officials to populate the list in the MainActivity onCreate.
    f. Add the onClick methods. The onClick can open a Toast message for now.
    g. Zip/City options-menu item opens the dialog, on entry you can open a Toast message for now.
    h. Create the About Activity – opens when MainActivity "About" options menu item is selected.

2) Add the location code.
    a. Add the location code that determines the device's zip code (this happens in onCreate).
    b. Instead of using that zip code to make the Google API call, you can open a Toast message that displays the zip code for now.

3) Add the Official Activity:
    a. This activity opens when an entry in the list of elected officials is clicked on.
    b. You can use the data in your test (dummy) official objects to test this activity.
    c. Any test data you don't have (i.e., missing data) should be properly handled in your activity.
    d. The social media ImageView onClick's should open a Toast message that displays the name of the social media for now.
    e. Remember to create the separate layout for landscape orientation.

4) Add the Google Civic Information API elements:
    a. Create the Google Civic Information API AsyncTask.
    b. Remove the use of dummy data from your app (now you will have real data).
    c. Remove the location Toast message. Instead, here you use the device's zip code (or a manually entered location) to make the API call.
    d. This should result in a populated list of Official objects in your MainActivity that is then displayed in the RecyclerView.
    e. Add the Photo Activity, opened when an official's photo is clicked in the Official Activity
    f. Remember to create the separate layout for landscape orientation.

5) Test the app very thoroughly and review your implementation against all requirements.

Extra Credit

Up to 20 points of extra credit will be awarded if you implement Up/Home Navigation (as previously discussed in class) in the Official Activity, the Photo Activity, and the About Activity. Up Nav should bring you back to the Main Activity. Points awarded depend upon the correct implementation of this feature.

Assignment Assistance

The TAs for our course is available to assist you with your assignment if needed. Questions on assignment requirements and course concepts can be sent to the instructor.

Submissions & Grading

1) All submissions *will* be graded once the due-date arrives. NO resubmission can be made at that point, regardless of the excuse. If you are not ready to be graded, DO NOT SUBMIT.
2) All submissions must conform to all requirements in this document, no exceptions. Please be sure to carefully review this document against your implementation to insure you did not miss anything.
3) Submissions must consist of your zipped project folder *(please execute Build =>Clean Project before generating the zip file*).
4) Submissions should reflect the concepts and practices we cover in class.
5) Grading will be based upon the presence and proper functionality of all features and behaviors described in this document.

NOTE

This assignment is worth 400 points. This means (for example) that if you get 89% on this assignment, your recorded score will be:

(89% * 400 points = 356 points)

If you do not understand anything in this handout, please ask.

Otherwise the assumption is that you understand the content.

Unsure? Ask!