Challenge #2

Project Group 1: Challenge #2

Michael Rzepka (34) (SG-11)

Haritha Garikapati (14) (SG-11)

Wang Zhang (50) (SG-3)

Ran Chen (11) (SG-3)

Dr. Yugyung Lee

CS 551: Advance Software engineering

October 26, 2014

**2) Features Implemented**

**a) Web Services**

**i) Custom Services**

A custom REST Web Service was created which stores the ID of any object that has already been captured by the system. This simple service allows history tracking based on the ID of the value passed in. The use case for this custom REST Service was incorporated with the Instagram functionality explained in more detail below. Essentially, an Image ID (returned from that service) is passed to an external REST service which will return a boolean indicator of whether that Image has previously been encountered or not. This in turn was used to populate the “Seen” indicator in the Instagram portion of the application.

The Google Maps Api for android V2 has also been used to search, find and locate the places on earth. We can simply input the location required and view the maps. It has been developed using the Google Play services Library. The user’s location is determined using WiFi and mobile cell data or by using GPS. The OpenGL ES V2 Required for Google Maps V2 has been used along with write to external storage as google maps store map data in external storage. Sometimes the app is being crashed despite all code being correct.

Another custom REST web Service was created which stores the user sign up information. Such as username, password, first name, last name, phone number and date of birth. Initially This Rest Web Service used by our application of registration part, but there is error which refuse app to connect to Rest Web Service.

Another custom Soap web Service was created to stores the user personal information. such as first name, last name, age, school name, major, address and phone number. initially, this Soap web service was implemented as web app and tried to display it in our Android app as web view, but it has same problem as last Rest web service. So it only can run it as web application.

**ii) Existing Services**

For the Instagram API, we incorporated a portion of the application to function as an image searching tool. The app allows a user to type in a “tag” (search keyword) that is then searched against the Instagram API of recently added images. The images found are then displayed to the user, allowing them to cycle through the images as they desire. One of the most important, desirable, aspects of this application is that the user can do all of this searching against Instagram without needed any authentication or account (which is not built into the Instagram website). The application also displays a “Seen” indicator, to determine if the image show has already been viewed by the user, using the custom web service as previously mentioned.

For the QR Code API, the user would be able to generate the QR code with color and any text they want.

**b) Database**

**i) Custom Data Model**

A very simple custom database was built behind the ID tracking custom REST Service that was created. It simply allows the insertion of the ID into a tracking database. This database is built externally from the application (as opposed to using the app’s local database), which would allow tracking this information across devices (as opposed to being localized to a single device).

A user registration database was built for login and registration custom REST Service that was created. it uses to store the user registration information and verify the user login account.

A user personal information database was built to store user information.

**c) Mobile User Interface**

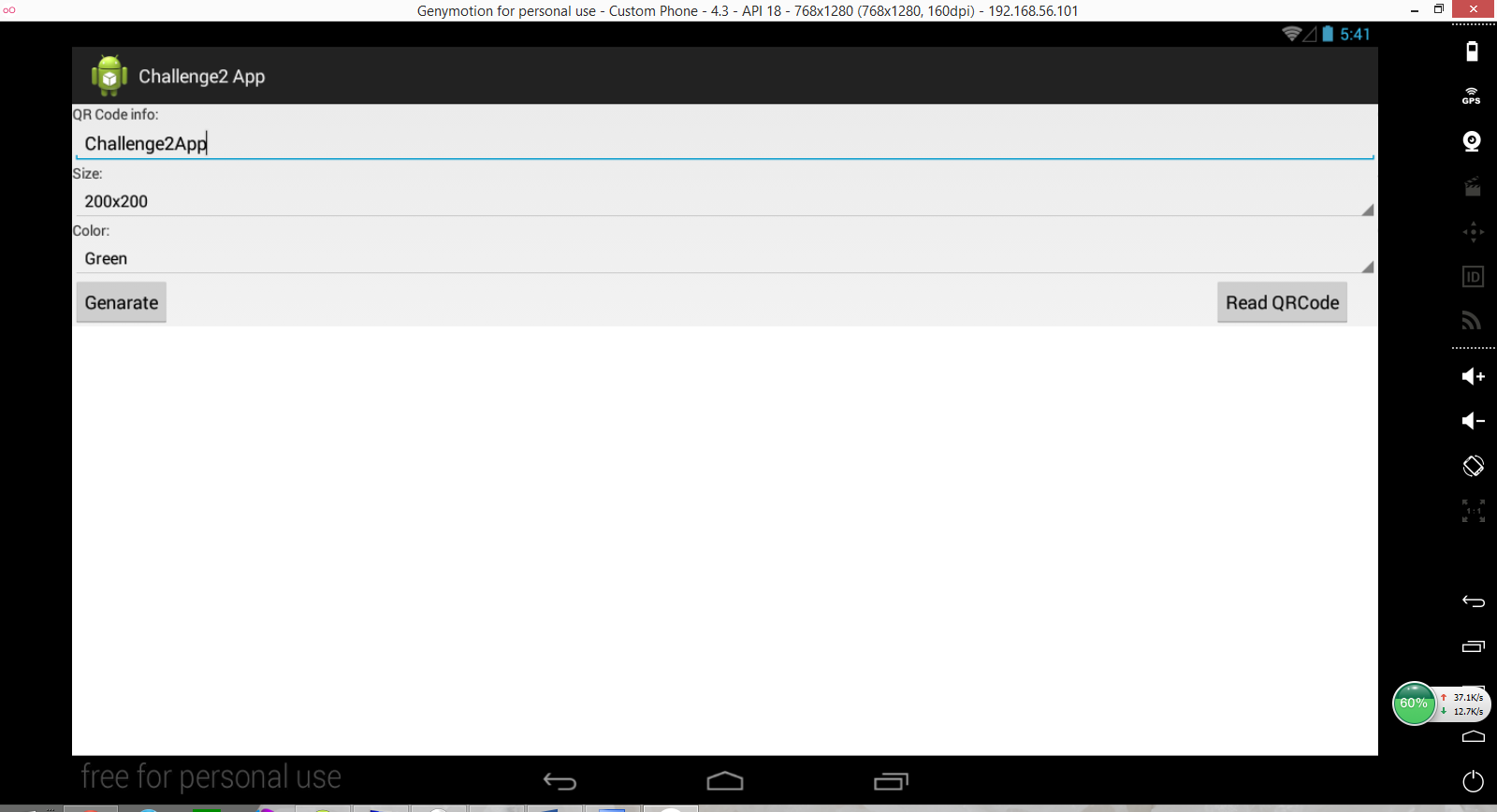
**i) Custom Interface**

For the Instagram portion of the application, a dynamic image view was used. This required some custom codification, which would dynamically reference the image URL from the tagged images returned from the Instagram API. Once the image is returned, it’s placed within an ImageView within which the image is drawn. There is also a next button to browse through a series of images, and a “Seen” indicator, as already explained above, which will populate if the image has been viewed. There is a tag search text box at the top to allow for searching of images.

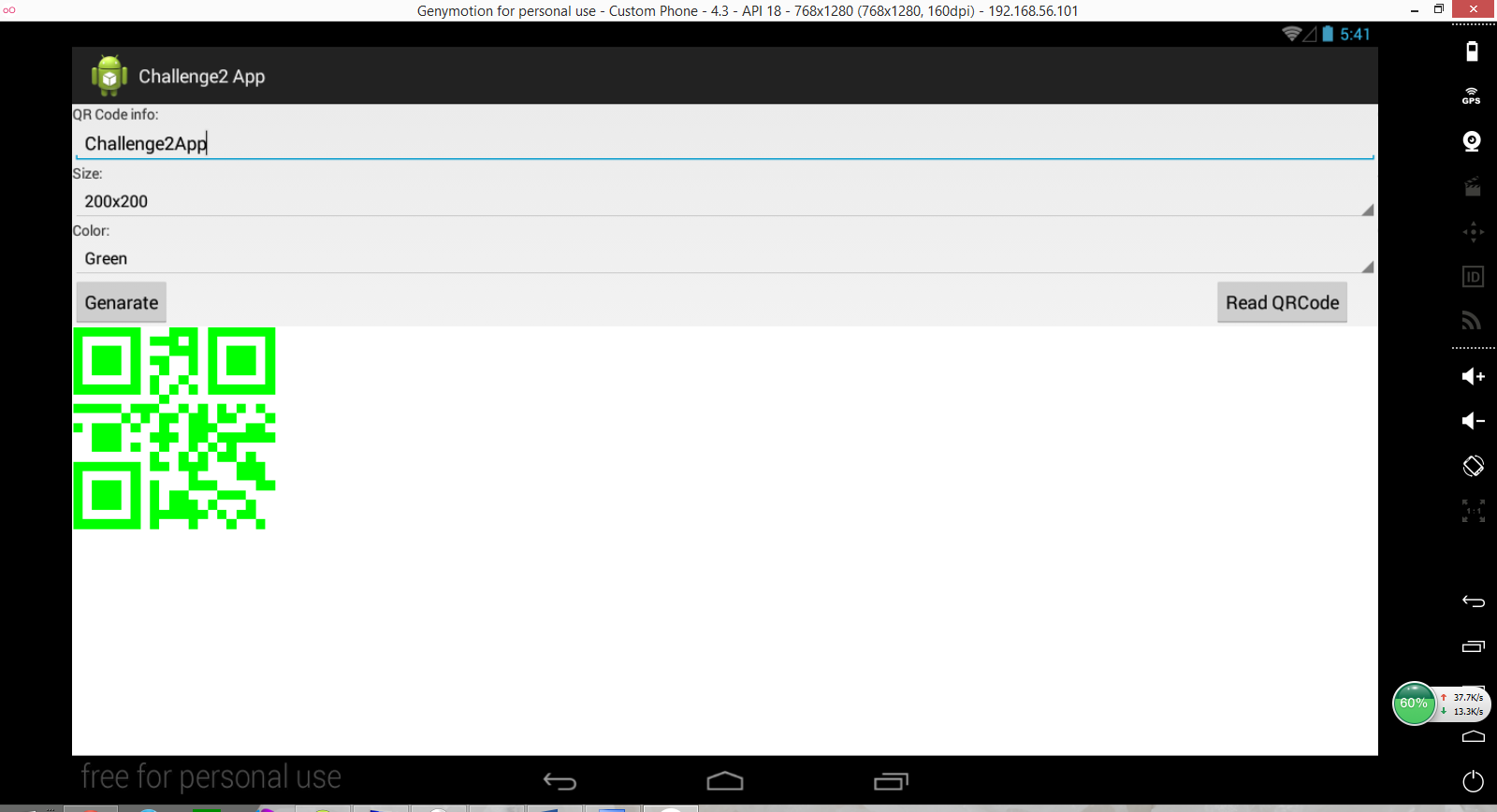
For the QR code portion of the application. We used QR code API. User need to enter the information they want to display in QR code and choose the size of QR code, also user would able to choose the color they want to generate for QR Code.

For the personal information portion of the application. it can store the user information. Also it can query information which stored in database and delete the user information from the database.

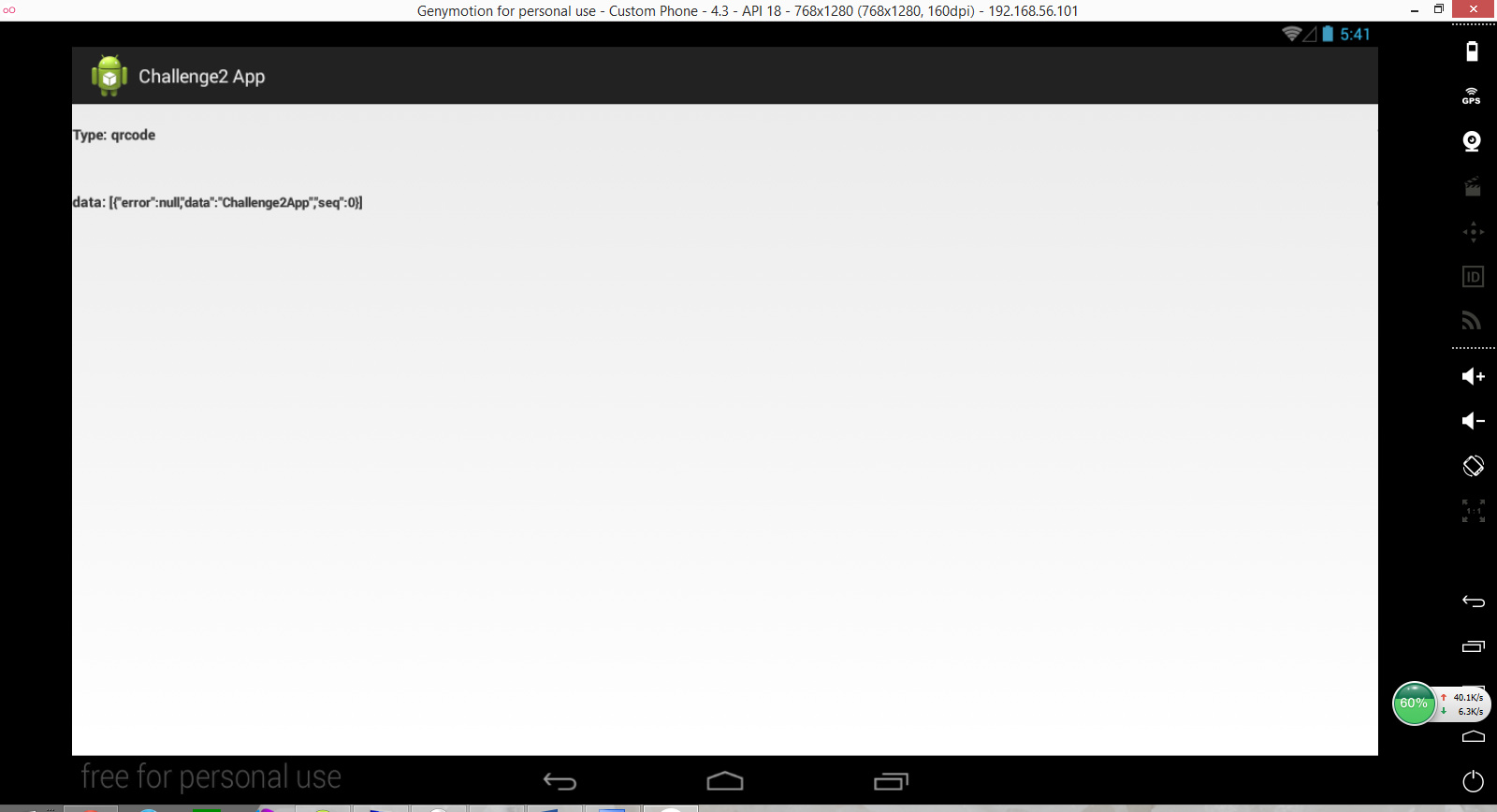
**3) Screenshots of Features**



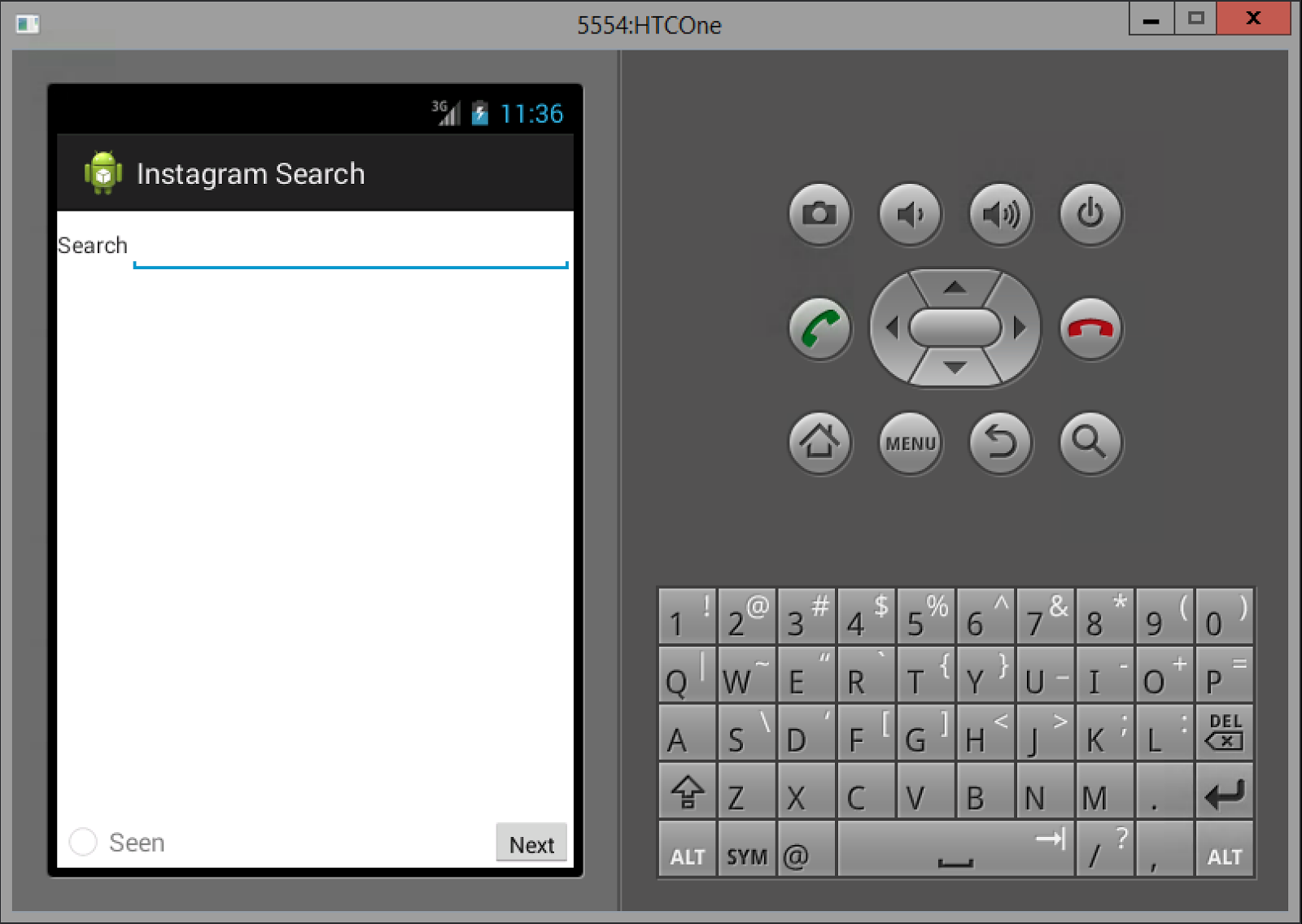
Main view of QR code portion.



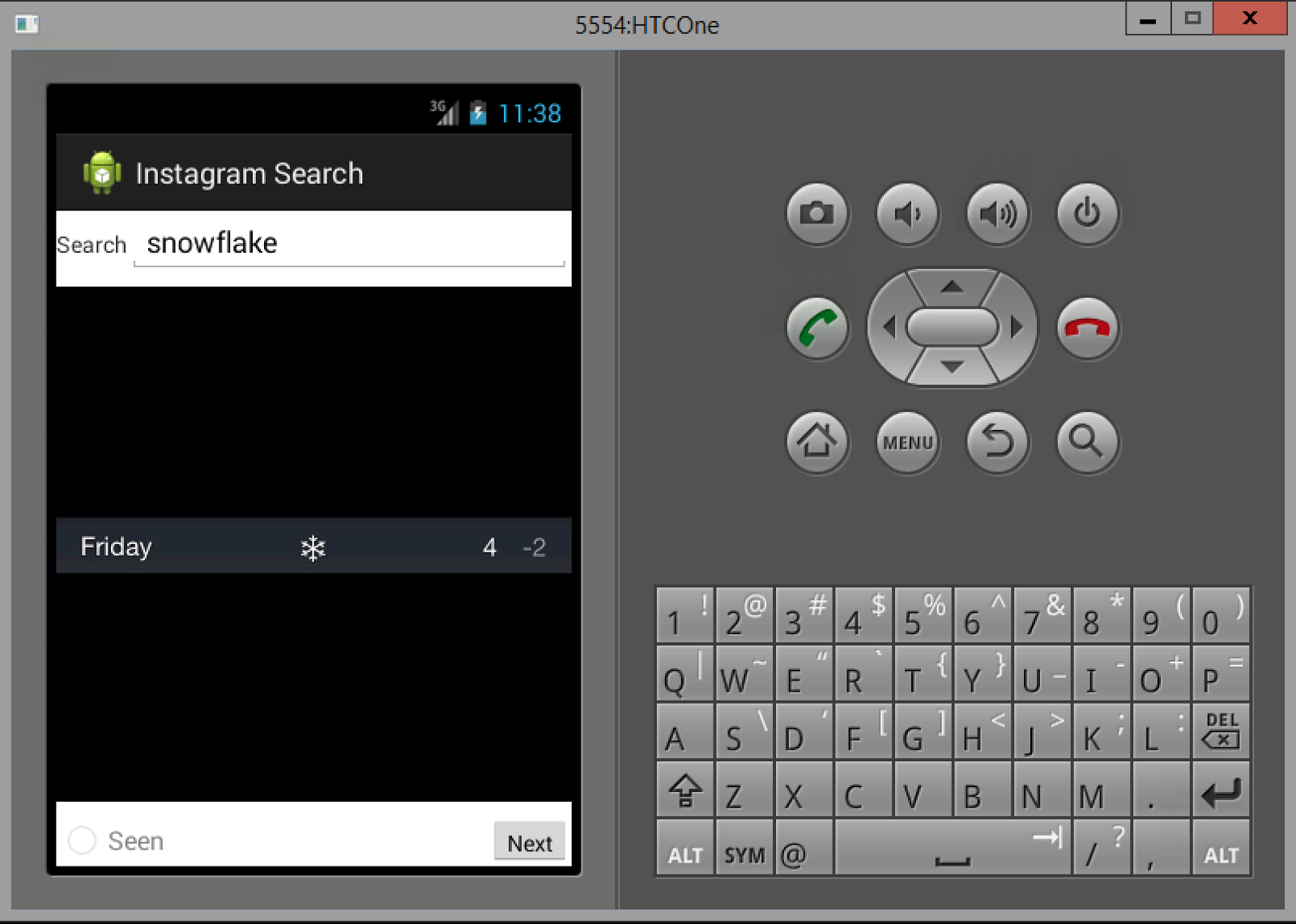
after generate the QR Code



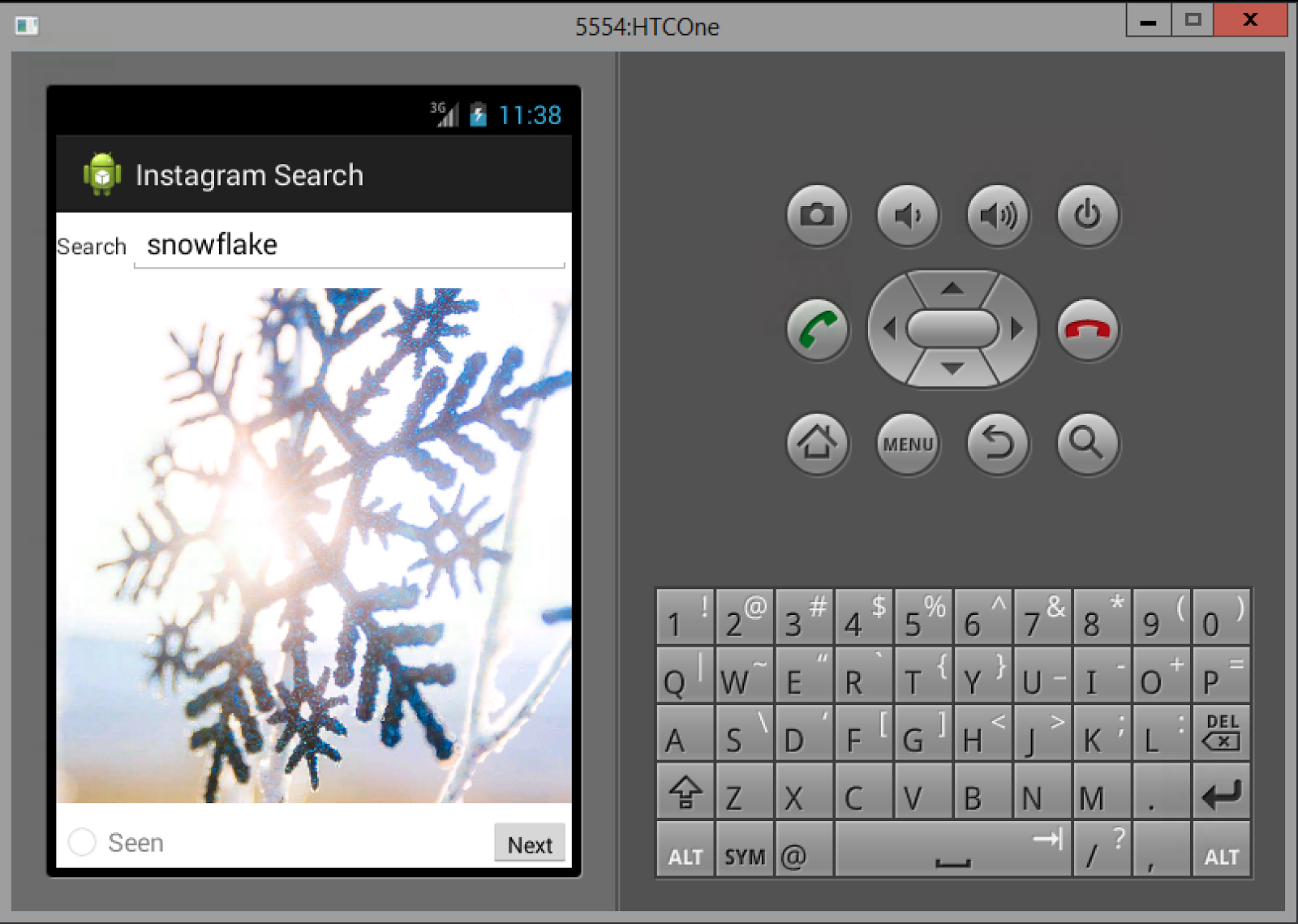
view after click read



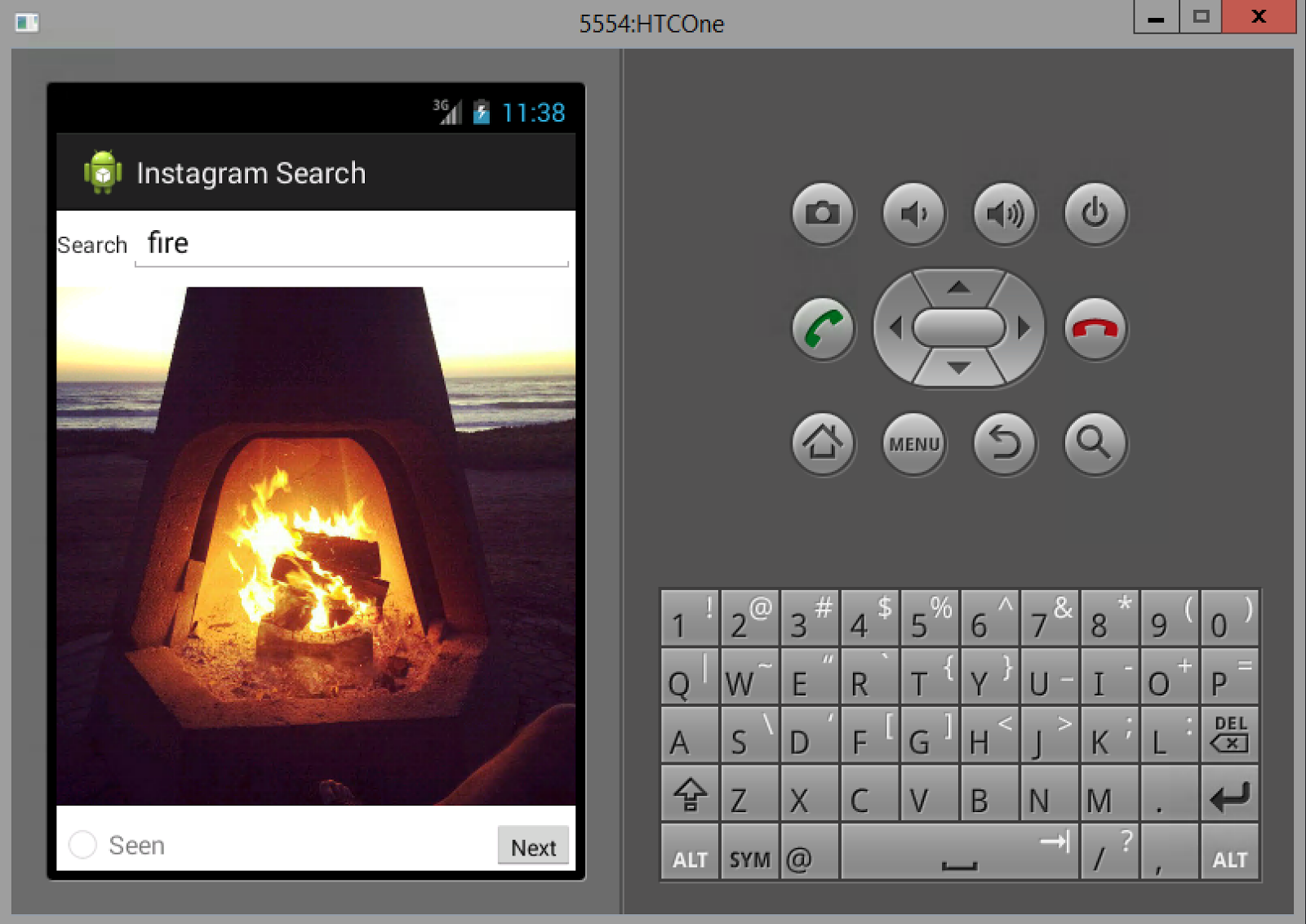
The Instagram feature’s initial state



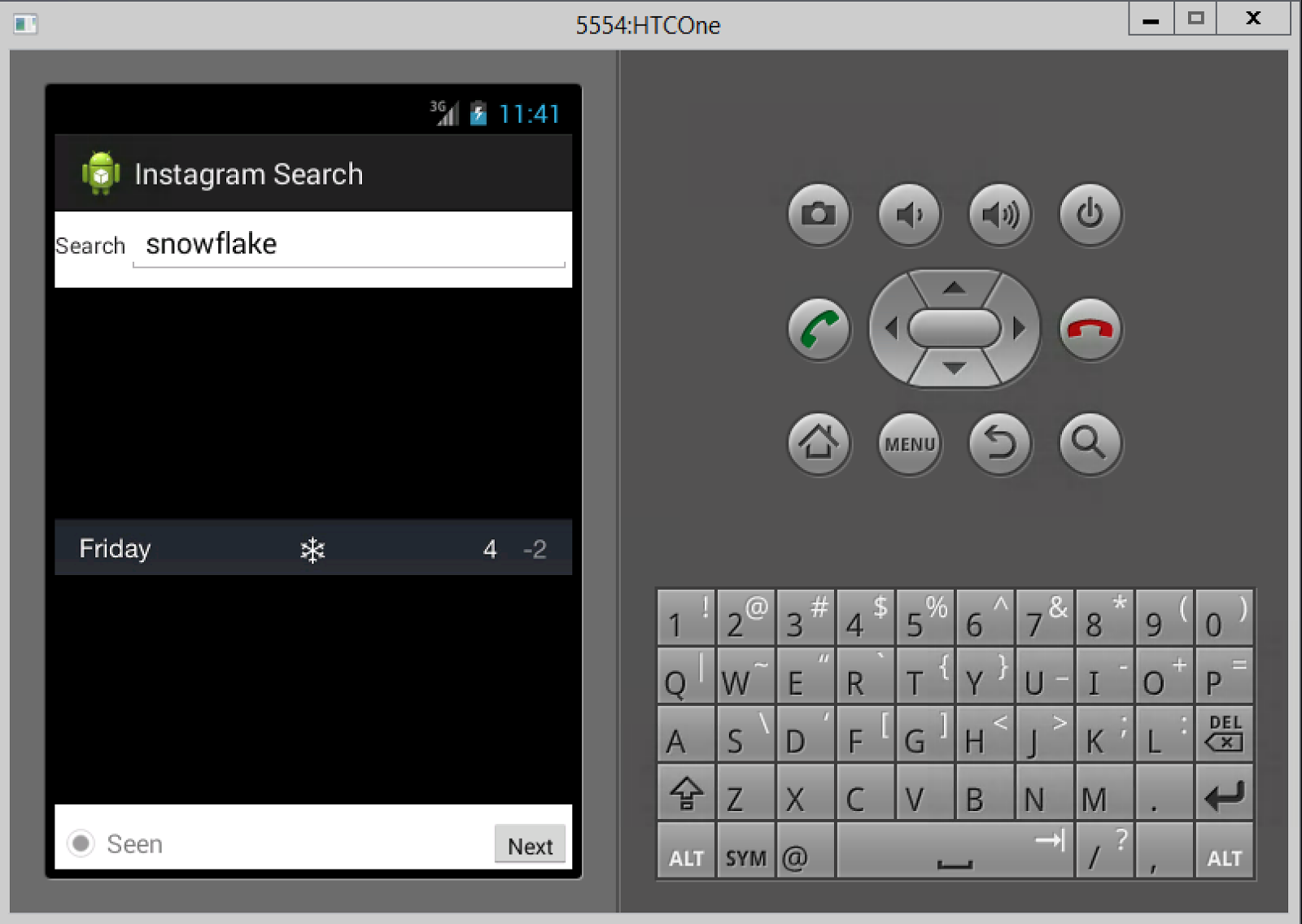
The Instagram feature’s tag searching



The Instagram feature’s image cycling

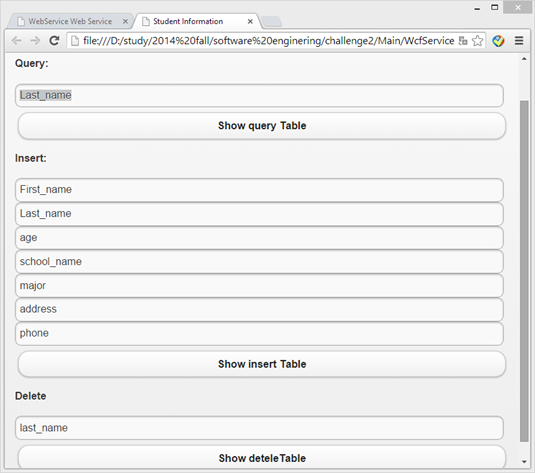


The Instagram feature’s tag changing functionality

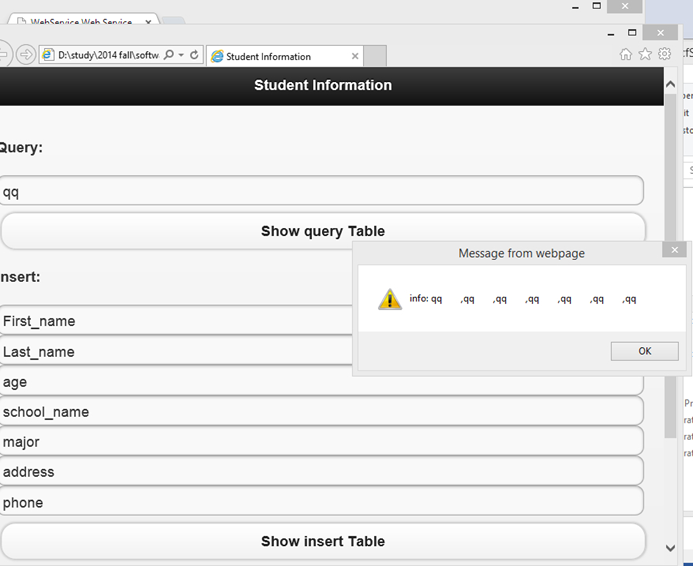


The Instagram feature’s seen indicator functionality

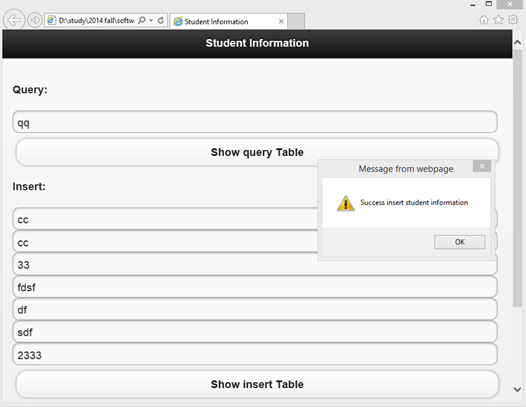




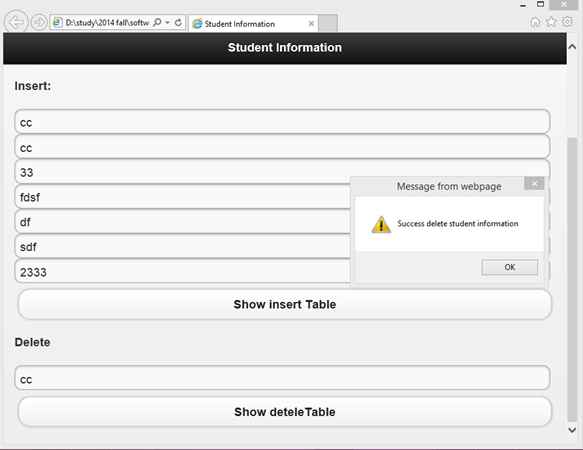
web application view



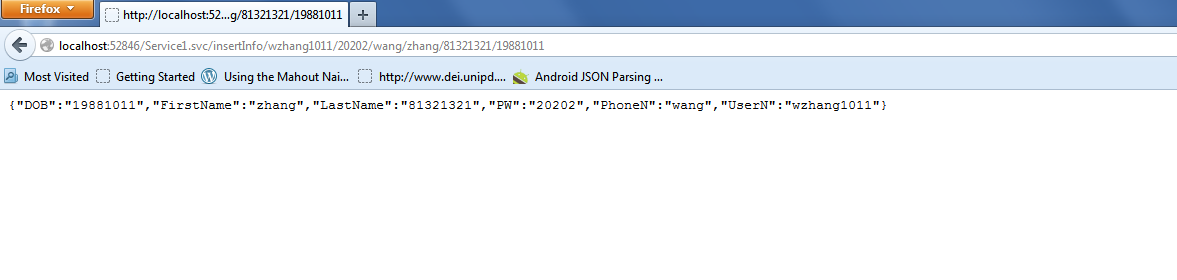
query table vew



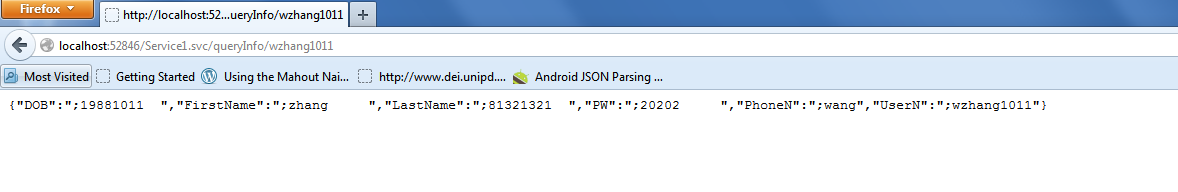
insert table view



Delete table view



Restful web Service for insert table with use to store user registration information



Restful Web Service for query table with use to verify the user login account

**4) Web Service URLs**

Instagram API: <http://instagram.com/developer/endpoints/tags/#get_tags_media_recent>

QR Code API:<http://goqr.me/api/>

Google Place API: <https://developers.google.com/places/documentation/?csw=1>

**5) Github URL**

<https://github.com/mtrzepka/ase-challenge2>

**6) Limitations**

Communication efforts were a very real limitation to this Challenge (probably more so than the first challenge). The reason for this is that more people were integrated into the project. Even though the duration of how long the project could run was extended, the added coordination between more people is challenging to overcome without a fixed strategy.

Another core limitation faced was that only one person was allowed to make changes to our designated remote server. This was an extreme bottleneck, as the group is dependent on this one person to be able to incorporate these components. Having more group members have access to this functionality (even if not all of them), would have greatly helped ease this burden.

**7) References**

Download Image Task: <http://stackoverflow.com/questions/2471935/how-to-load-an-imageview-by-url-in-android>

JSON Parser: <http://stackoverflow.com/questions/19746800/android-parse-jsonobject>

Http Handler: <http://www.androidhive.info/2012/01/android-json-parsing-tutorial/>