# Project 4 Task 2 – Dictionary App

**Nitish Mudgal**

## Description:

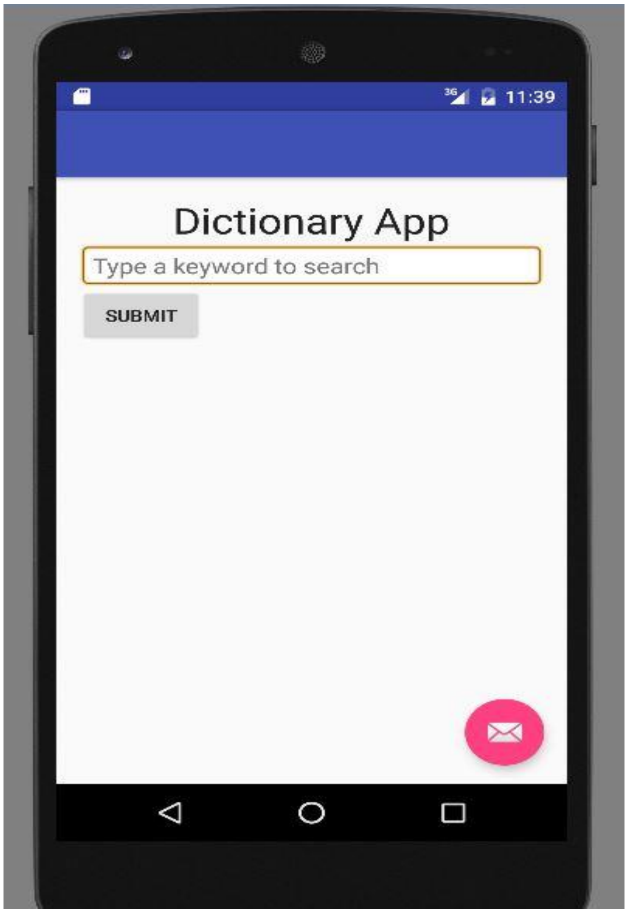
My application takes a search string from the user, and uses it to fetch and display the meaning of the string and the synonyms, antonyms associated with it. The application calls the web service hosted on the cloud which in turn calls the Merriam Webster API.
If the API doesn't return any meaning of the search term, my app shows the suggestions of the word.
Here is how my application meets the task requirements:

1. Implement a native Android application

The name of my native Android application project in Android Studio is:
Dictionary

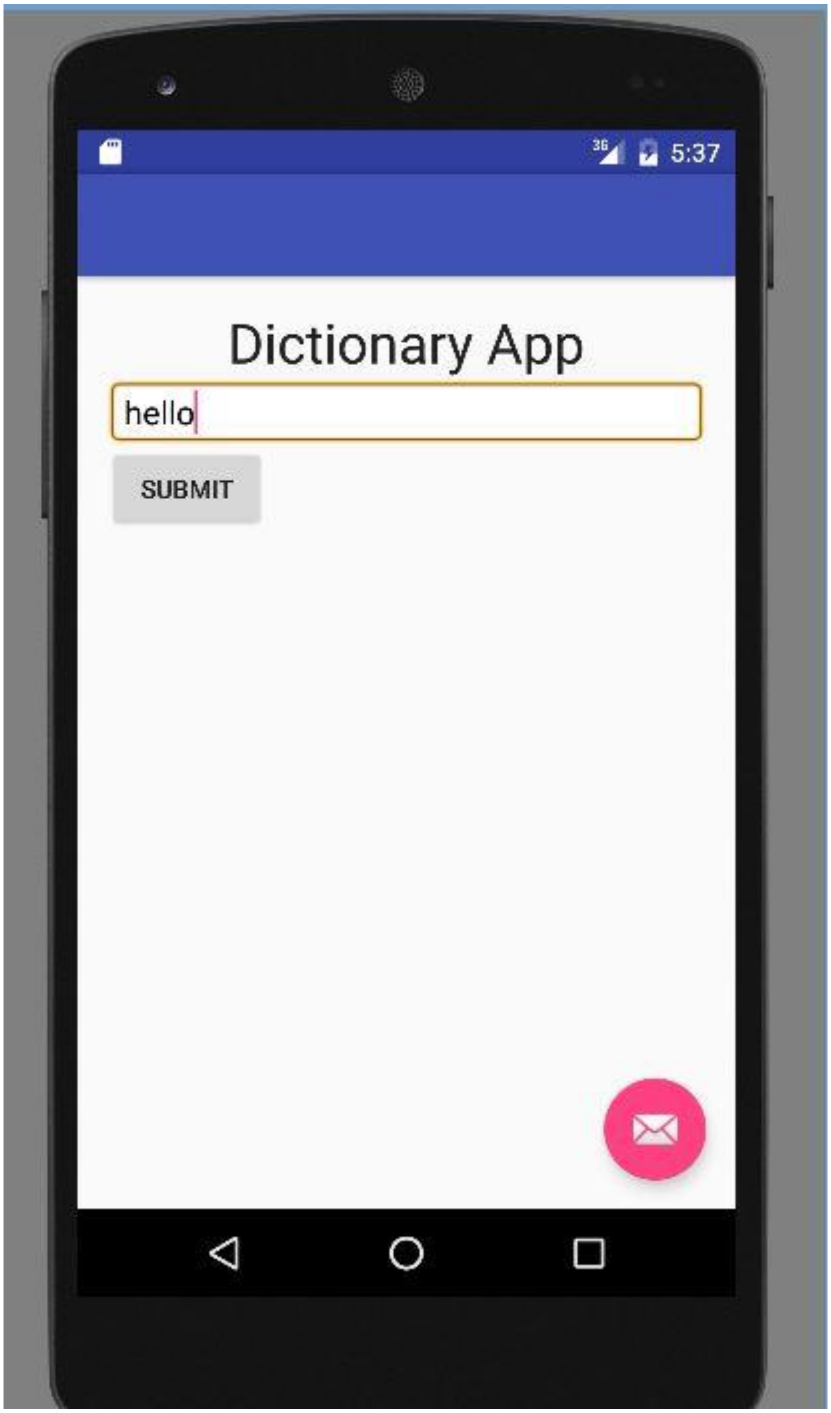
1.1. Has at least two different kinds of views in your Layout (TextView, EditText, ImageView, etc.)

My application uses two TextView, one EditText, and a Button. See content_main.xml for details of how they are incorporated into the LinearLayout.
Here is a screenshot of the layout before the picture has been fetched.



## 1.2. Requires input from the user

Here is a screenshot of the user searching for meaning of “Hello”.



## 1.3. Makes an HTTP request (using an appropriate HTTP method) to your web Service

My application does an HTTP GET request in GetDictionary.java. The HTTP request is:
**https://warm-chamber-91456.herokuapp.com/servlet?searchstring="** + searchTerm
where, searchTerm is the user's search term.
The search method makes this request to the web service deployed in Heroku, where the web service parses the xml response from Merriam Webster REST API and sends it back to the android application where it is displayed.

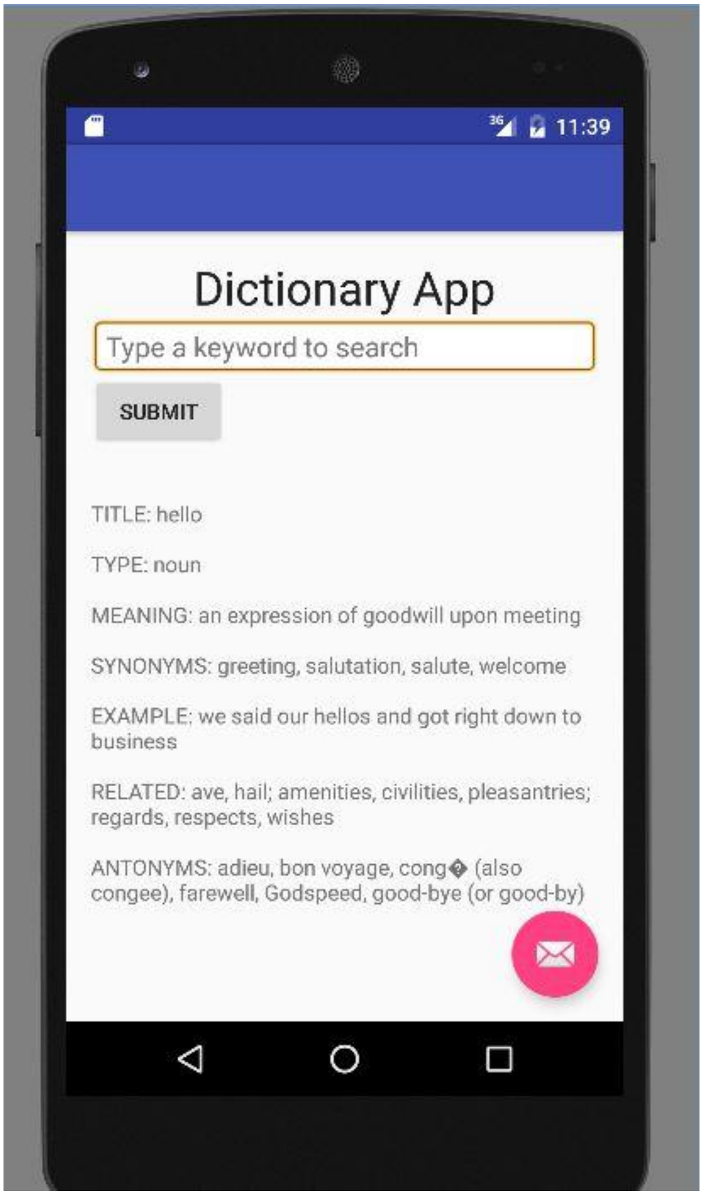
## 1.4. Receives and parses an XML or JSON formatted reply from the web service

An example of the XML reply is:

```
<synonym>
      <title>hello</title>
      <type>noun</type>
      <meaning>an expression of goodwill upon meeting</meaning>
      <synonyms>greeting, salutation, salute, welcome</synonyms>
      <example>we said our hellos and got right down to business</example>
      <related>
            ave, hail; amenities, civilities, pleasantries; regards,
            respects, wishes
      </related>
      <antonyms>
            adieu, bon voyage, congé (also congee), farewell, Godspeed, good-
            bye (or good-by)
      </antonyms>
</synonym>
```

### 1.5. Displays new information to the user

Here is the screen shot after the response xml of word “hello” has been returned.



TITLE: hello

TYPE: noun

MEANING: an expression of goodwill upon meeting

SYNONYMS: greeting, salutation, salute, welcome

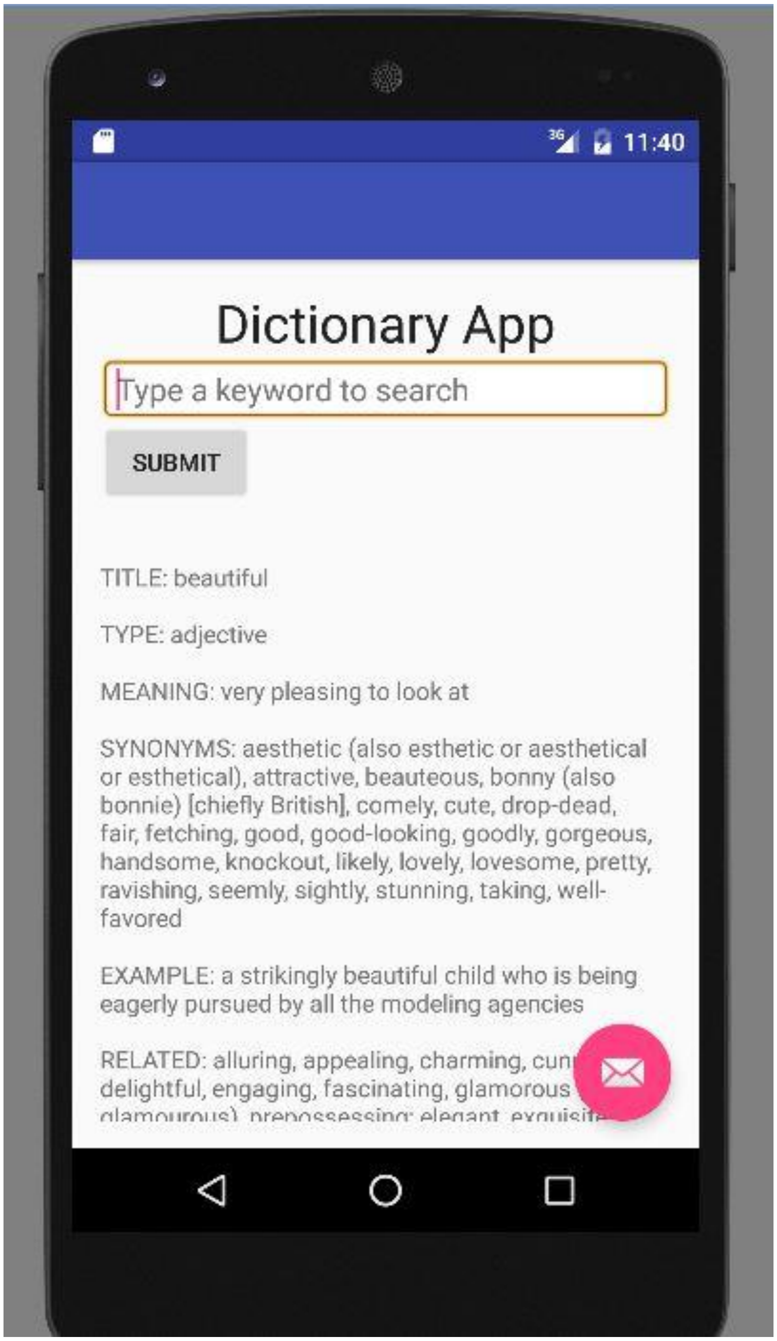
EXAMPLE: we said our hellos and got right down to business

RELATED: ave, hail; amenities, civilities, pleasantries; regards, respects, wishes

ANTONYMS: adieu, bon voyage, cong� (also congee), farewell, Godspeed, good-bye (or good-by)

1.6. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

The user can type in another search term and hit Submit. Here is an example having typed in "beautiful".



TITLE: beautiful

TYPE: adjective

MEANING: very pleasing to look at

SYNONYMS: aesthetic (also esthetic or aesthetical or esthetical), attractive, beauteous, bonny (also bonnie) [chiefly British], comely, cute, drop-dead, fair, fetching, good, good-looking, goodly, gorgeous, handsome, knockout, likely, lovely, lovesome, pretty, ravishing, seemly, sightly, stunning, taking, well-favored

EXAMPLE: a strikingly beautiful child who is being eagerly pursued by all the modeling agencies

RELATED: alluring, appealing, charming, cun
delightful, engaging, fascinating, glamorous
glamourous) prepossessing; elegant, exquisite

### 2. Implement a web application, deployed to Heroku

The name of the web application deployed on Heroku  is: NetBeansTask2WebService

### 2.1. Using an HttpServlet to implement a simple (can be a single path) API

In my web app project:
Model: Model.java
Controller: Servlet.java
POJO: Synonyms.java

### 2.2. Receives an HTTP request from the native Android application

Servlet.java receives the HTTP GET request from the Android application with argument "searchstring". It passes this search string on to the model.

### 2.3. Executes business logic appropriate to your application

Model.java makes an HTTP request to Merriem Webster Restful API:
http://www.dictionaryapi.com/api/v1/references/thesaurus/xml/ + searchString + key
It then parses the XML response and extracts the parts it needs to respond to the Android application.

### 2.4. Replies to the Android application with an XML or JSON formatted response.

It then creates a xml doc and sends the response to the android application using Document Builder class and Factory.