1-1-2013

An Application for Keeping Track of Food Item Expiration

Rejin Paul James
University of South Carolina - Columbia

An Application For Keeping Track Of Food Item Expiration

by

Rejin Paul James

Bachelor of Engineering
Karunya University, 2011

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2013

Accepted by:

Michael Huhns, Director of Thesis

Marco Valtorta, Reader

Yan Tong, Reader

Lacy Ford, Vice Provost and Dean of Graduate Studies

DEDICATION

I would like to dedicate this thesis to my parents, brothers and friends. Thank you very much for all of your support and encouragement while I wrote this thesis.

ACKNOWLEDGEMENTS

At the outset, I would like to express my deepest gratitude to our Lord, the **GOD ALMIGHTY**, for his abundant grace and guidance without which this thesis would not have been taken up and successfully completed .

I would like to place my heartfelt thanks and gratitude to my thesis advisor **Dr. Michael Huhns,** I am indebted to my advisor for his valuable support, advice and encouragement and for the overall guidance that he has provided to me for the successful completion of the project.

I also thank my thesis committee members **Dr. Marco Valtorta** and **Dr. Yan Tong** without whose assistance this thesis would not have been successful. With gratitude I remember the support and encouragement of my parents and friends who have prayed and helped me a lot during my thesis work.

ABSTRACT

Food, honestly, is too precious to waste. Food wastage is a very serious issue prevalent in the world today. American households alone throw out an equivalent of $165 billion worth of food each year. People often forget to consume food they purchased before the expiration date, or sometimes they over-purchase food they can have, then throw them away. Hence, this thesis aims to prevent food wastage with the help of a smart phone application that helps keep track of food item expiration dates and gives you notification alerts when it is about to expire. It implements a barcode scanner for automatic product name discovery as well as optical character recognition (OCR) for automatic food expiration discovery.

TABLE OF CONTENTS

LIST OF FIGURES

List of Abbreviations

API ..... Application Programmable Interface

EAN ..... European Article Numbering

GUI ..... Graphical User Interface

IDE ..... Integrated Development Environment

JSON ..... JavaScript Object Notation

NDK ..... Native Development Kit

NRDC ..... National Resources Defense Council

OCR ..... Optical Character Recognition

SDK ..... Standard Development Kit

UPC ..... Universal Product Code

ZXing ..... Zebra Crossing

CHAPTER 1

INTRODUCTION

Food wastage due to expiration is an issue that is evident in almost every American household and needs to be addressed. It has been ascertained by a study conducted by National Resources Defense Council (NRDC) that 40 percent of food in the United States today goes uneaten. This means that more than 20 pounds of food per person every month goes uneaten. Not only does this imply that Americans are throwing away $165 million each year, but also 25 percent of all freshwater and enormous amounts of chemicals, energy and land. It is estimated that if the United States wasted 5 percent less food, it would be enough to feed 4 million Americans.

If we consider only American households, it is estimated that approximately 25 percent of food bought is wasted. Food wastage has serious implications for wasted energy. Anecdotal evidence suggests that the following are the major factors that drive household losses -

i. undervaluing food
ii. unideal storage
iii. misjudged food needs
iv. partially used ingredients
v. bulk purchases
vi. poor planning, and
vii. over preparation

It has been proven by a study that household food wastage is less in developing countries as compared to developed countries. The study then implies that the average American consumer throws 10 times as much food as the average southeast Asian consumer.

Hence, a smart phone application has been proposed by this thesis, which will help keep track of food expiration dates and notify the user when a food is about to expire so that food wastage is prevented. This application will enable the user to scan the barcode of the food product to automatically discover the product's name. The user will then be able to use the optical character recognition feature to capture the date of expiry from the product, after which the user can add the item to a database and the date is added to the calendar which will then enable notifications to pop up before food is about to expire.

This project is novel because it tries to implement an optical character recognition module to identify the expiry date, which has never been used in this kind of application before, along with a barcode scanning module, which has just been used in one Android and one iOS applications of this kind. There has never been a combination of both these features in any application of this kind. The combination of both the modules may save a significant amount time spent during item entry.

The definitive objective of this project is reducing food wastage by developing an application to track food expiration. This application will be developed with a consideration that it does not take up significant time of the users to feed the food item details into the application.

CHAPTER 2

MOTIVATION

The motivations behind developing such an application are the following:

i. Economic Benefits

- Reduce over-purchasing - Americans throw $165 million worth of food each year. We can save a significant amount of money if the food we purchase is not wasted when we use it before it expires. It may also deter us from over-purchasing.

- Lower disposal costs - If each household decreases food wastage there may be an overall reduction in the food disposal costs on the economy.

ii. Social Benefits

- Feed people - By keeping track of when the food is going to expire, it prevents wastage and if we have surplus food we can donate it to food rescue organizations.

iii. Environmental Benefits

- Reduce resource use essential to food production - Many resources are needed to grow food such as water, pesticides and energy. By wasting food, these resources are also wasted that went into growing it

- Reduce Landfills - By reducing food wastage we also in turn reduce Landfills which are a major source of the greenhouse gas Methane.

CHAPTER 3

BACKGROUND

The idea of using a smart phone application to save food item expiration dates has been employed before. An application which provides the manual entry of item names and dates has been developed before in Android [7,8] as well as in iOS [9]. Even though these applications save the details of the food in a database and give notifications, it takes a really long time to manually enter each item into the application and may be inconvenient to a typical user.

So, significant improvements on such kind of applications were made to reduce the time taken by manual entry. One of the improvements was implementing a bar code scanning feature to scan the barcodes of food products which has been developed in both Android [5] and iOS [6]. Both these applications do improve the time taken to enter each product. The Android application does not include color coded display to highlight the food items that are close to expiry and the ones which are not near expiry.

All the previous work mentioned does have certain improvements but the proposed thesis aims to save some more time by implementing an Optical Character Recognition module that will help scan the expiry dates on the products to automatically discover the expiration dates. It also provides an alternative so that the date can be entered manually as well. In addition to this, a color coded scheme is used to list out the food items so that it highlights the food items close to expiry.

CHAPTER 4

PROBLEM STATEMENT

Though the problem of food wastage has been quite prevalent in the past few years, significant efforts were not made to prevent it. As observed by recent trends , American households tend to ignore the issue of food wastage and do not take necessary steps to avoid it. This contributes a huge chunk to the total food wasted by the United States.

It has been proven by a study that household food wastage is less in developing countries as compared to developed countries. The study then implies that the average American consumer throws 10 times as much food as the average southeast Asian consumer.

As discussed in the background, the existing applications are not adequate enough for the current problem, so this thesis aims to improve on their shortcomings and thereby provide a complete solution.

CHAPTER 5

PROPOSED SOLUTION

As mentioned above in the problem statement, a comprehensive solution is needed to tackle food wastage and targeting households will significantly reduce a large amount of food that is wasted. The application needs to be very easy to use as well as it needs to significantly reduce user effort and time. Current applications do not implement OCR which may save a considerable amount of time in data entry. They also do not display the list of items efficiently.

For this thesis , the design of the application would then consist of a Graphical User Interface (GUI) where the user can view the current list of items that is already present in the database. In order to add a new item the user will have options such as scanning the barcode of the product, scanning the expiry date of the product and finally adding the product to the database to get the notifications.

5.1 COMPONENTS

The components of this application (see Figure 5.1) are as follows:

i. The Application with GUI

   This is the main graphical user interface of the application. When the application is launched, the initial interface consists of a view that shows the list of items that are already added in the database. The list of items will be sorted in the

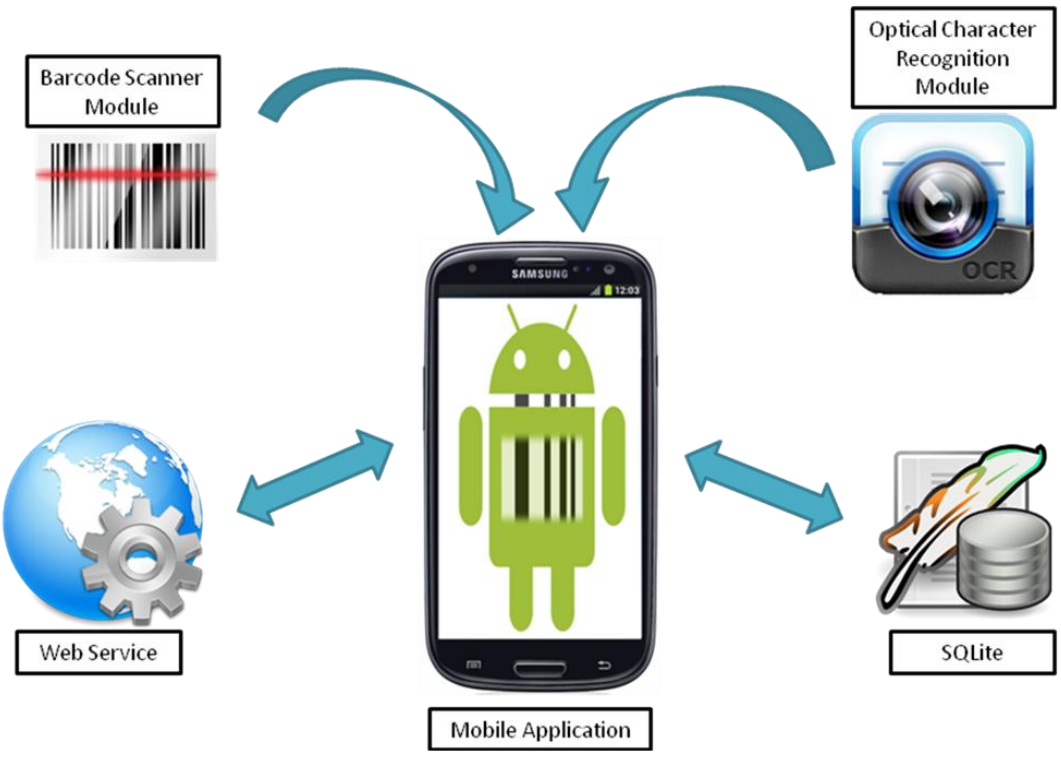

Figure 5.1 Design Components

order of closest date of expiry and will also have color bar indicators that will make it easier to know which items are high priority. There will be a settings option where we can set different preferences of the application. The preferences will include the option to enable or disable notifications, setting the notification time each day to remind what food is expiring, set the number of days before an item expires that the notification will show. There will also be an option of auto removal of items once they expire and the number of days after expiry that an item should be removed from the list as well as the database.

In addition to this, there will be an option in the view to add new items to the list which when clicked on will open a new view where the details of the new item can be filled out. This view will have buttons that will help users to

access other modules of the application such as the barcode scanner and the optical character recognition module for automatic product discovery as well as automatic expiration date discovery respectively. Furthermore, it will also have the feature to manually enter the product name if the product name is not automatically discovered and to manually enter expiration date if date is not recognized by the OCR. There will also be a button to finally save all the information of the food item to the database.

ii. Bar Code Scanner Module

For scanning the barcode information on the back of the food products this project uses Google's ZXing (Zebra Crossing) multi-format 1D/2D barcode image processing library implemented in Java. This library is open source and very easy to integrate in the project but requires an additional application called Barcode Scanner to be installed on the phone. It is able to read both UPC (Universal Product Code) barcode format as well as EAN(European Article Numbering) barcode format, now known as International Article Numbering.

The authors of ZXing made it very easy for it to be integrated with other projects. It is achieved with the help of using Intents. Therefore, a barcode is scanned by calling the Barcode Scanner application via Intents. The authors provide a small library of code, which correctly handles all the important details, such as setting category flags and handling the case where Barcode Scanner application is not installed. This library of code makes sure to prompt the user to install Barcode Scanner application if it is not already installed on the smart phone as the ZXing library requires it, to work completely. The scanning result

is handled very easily by using some more code examples, given by the author, in the activity of the barcode scanner. The scanning result is then used by the JSON Parser module which contacts the web service to extract the name of the product.

The graphical user interface of the barcode scanner has a view that accesses the smart phone's camera. It overlays a selectable region in the view so that the barcodes can be comfortably scanned within the region. Once the barcode is scanned it displays a success message briefly along with a snapshot of the barcode. Then the main view where the item information is entered is pulled up.

iii. Web Service

A RESTful web service is used to automatically discover the product's name. The website which we use, that stores the database of barcodes, is http://www.upcdatabase.org. It is a project created by a small group of students in 2010 and includes barcodes in both UPC and EAN formats. It was started as a hobby and became popular very quickly. It provides access to the meta data of the barcodes like product name, manufacturer and price.

It provides developers with a JSON API service which can be used to conveniently query the database but it needs an API key which the developer gets once signing up on their website. A JSON parser class is used that has a method which queries the JSON API and does the parsing of the JSON data that it receives back. This method takes a URL as an argument. This URL belongs to the JSON API which has the API key appended to it as well as the barcode

number, that it receives from the barcode scanner module, appended to it. Once the JSON data is parsed to obtain the product's name, it is then display on the GUI. Since only the name of the food item is important for this application, the code is written such that it only extracts that.

It was possible that a private database of barcodes could have been used in this application, but it would have a major tradeoff that it would have been very large as it would have close to 2 million barcodes. Another reason that such a database was not used was because it would not have been convenient for the user to update their application each time the database was updated with new barcodes. Using the web service is more convenient since we do not need to update the application every time there are updates to the database that the web service provides. There is do not need to worry about updating the database as it is constantly updated by users and if a developer has to add a new item to the database, it is very easy to do so.

iv. Optical Character Recognition Module

This thesis uses the Tess Two library which is based on the Tesseract OCR Engine, developed by HP and now maintained by Google, and Leptonica image processing libraries. The Tess Two library is a fork of Tesseract tools for Android, which provides a set of Android APIs and build files for the Tesseract OCR and Leptonica, and adds some functions to it. It downloads the already trained language files when integrated with the application. The OCR module will capture the date of expiry of the product and display it on the screen after

some processing. The application then adds that date to the calendar on submission of the food item into the database.

The Tess Two library was readily available as an open source library which was used to integrate with the project. The major challenge was that the library was written in C/C++ language and in order to integrate it into the project the Android Native Development Kit (NDK) was required. Android NDK is a companion tool to the Android Software Development Kit (SDK) and makes it possible to port libraries written in C/C++ to Android. Since OCR does CPU intensive work the authors developed it using native code. The author of Tess Two also provided an example implementation of an Android project which was modified to suit this project's requirements.

The graphical user interface of the optical character recognition module has a view that accesses the smart phone's camera like in the case of barcode scanner. It also overlays a selectable region in the view so that the required date can be conveniently scanned within the region. Once the required date of expiry is in the small region, the camera button on the interface can be clicked to scan the date. If the scan results are accurate the done button can be clicked or if scan results are not satisfactory the skip button can be clicked to skip the scanning process. In this case, the main view where the food item information is added is shown and the date can be manually picked in the available date picker widget.

A pure Java OCR could have been used but on experimentation it was found that even after training it with a significant number of samples, it did not

give satisfactory results and hence the Tess Two OCR was preferred over it. In addition to it, already trained language files were available for Tess Two.

v. SQLite

SQLite is the database that is used if an application needs to manage its own private database. This database is used to store the list of items and their expiration dates. It's database management classes are readily available to store the application's content. Simple SQL queries can be used to query the database.

If the application was working on data sent to it by a provider then only the generic database classes would have been used. Android comes with the sqlite3 database tool. This tool can be used to browse or run SQL commands on the device. It is run by typing sqlite3 in a shell window. A separate preferences file is used to save the preferences of the application. The database will only consist of one table with a column each for id, name and date of expiry.

5.2 FLOW CHART

The flow chart of the application is given in Figure 5.2. It shows that when the application is launched it calls the main activity that has a view which displays the list of all the food items that are already added to the database. This view has the items in the order of closest day of expiry, so the item which is going to expire soon will show up on top of the list. In the options, the user has a choice to add a new item, change the preferences for the application and exit from the application. If the user chooses to add a new item, the AddItem view shows up which has all the options to add the necessary information related to the food item.

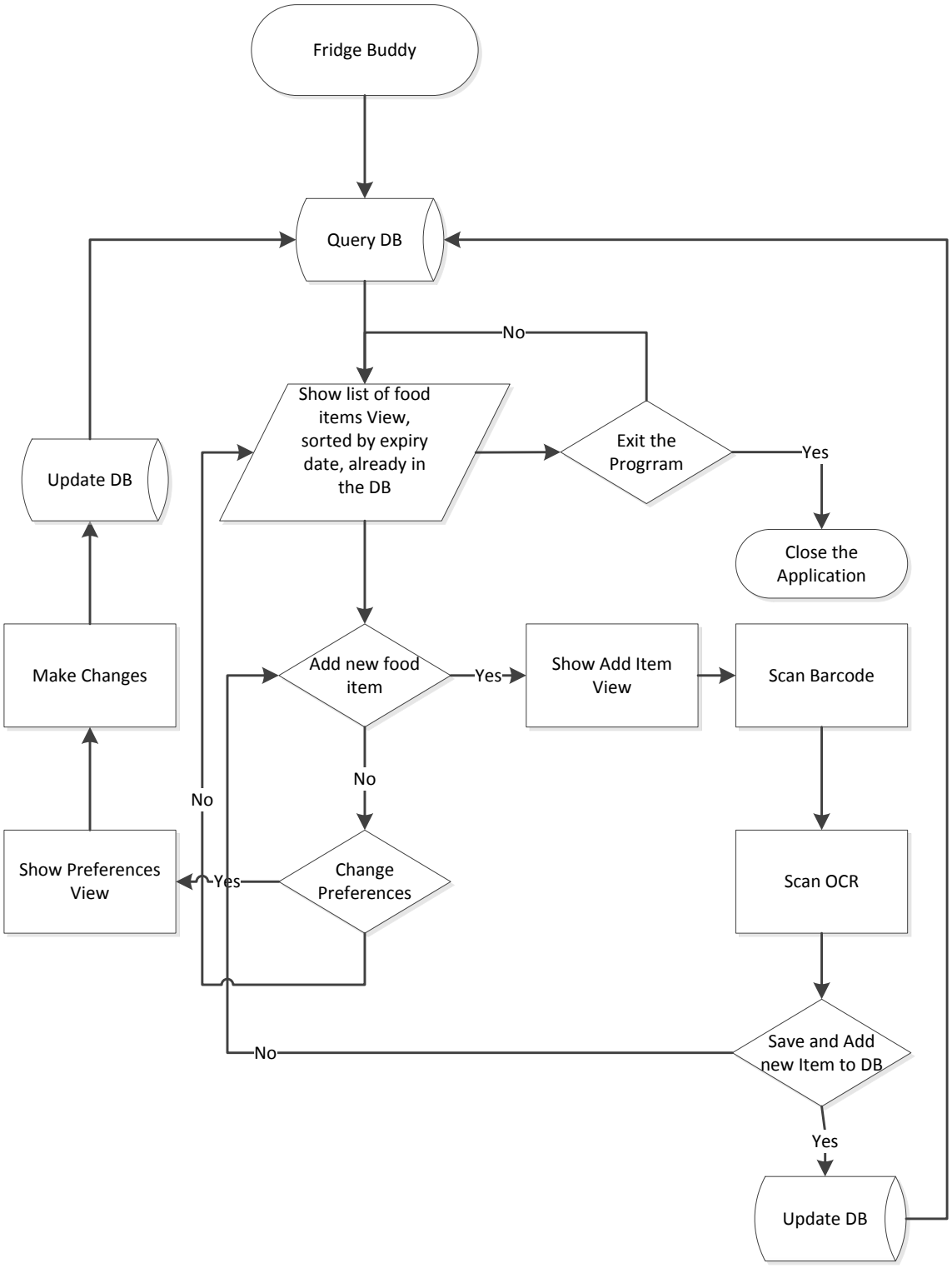

Figure 5.2 Flow Chart

First, the user can scan the barcode of the food item to be added by clicking the scan barcode button, which calls the bar code scanner module and opens up another view. This view accesses the smart phone's camera and overlays a selectable region in the view with a red horizontal line in the middle, so that the barcodes can be comfortably scanned within the region. Now the product with the barcode is placed in front of the camera so that the barcode is scanned, after which it displays a success message briefly along with a snapshot of the product with the barcode. Then in the background the application sends a JSON query, with the barcode number as a parameter, to the web service called www.upcdatabase.org. The web service processes the query and searches for product in its database and sends back a response. The AddItem view shows up with the name of the product if found in the database else it will prompt to add the name manually in the text field provided.

Secondly, the user then tries to scan the date of expiry of the product by clicking on the scan OCR button, which calls the OCR module and brings up another view. This view accesses the smart phone's camera and overlays a small selectable region in the view so that the dates can be scanned easily. The product's date is placed in front of the camera so that the date can be scanned. Once the date is scanned, the user can either click on a button called continue if he deems the scan result to be satisfactory or try to scan the date again. If the result is not satisfactory, they will also have a button to skip the OCR step, which will then bring back the AddItem view. Meanwhile, in the background the scanned date will be processed and then converted to the appropriate date format. The date picker widget will be updated with the new date. If the widget has not been updated,

then the user can manually set the date. Once all the information is complete the user can then click on the save button to add the item to the SQLite database and update it.

The main view is then shown again with the updated list. Now, if the user wants to change the application's preferences he can select the change preferences options in the settings. The change preferences view then comes up on the screen which has all the different preferences of the application. When we are done making the changes, we can press the back button which will then update the preferences in the preferences file. The main view will show up again from where we can click on the exit button in the settings to exit the application.

CHAPTER 6

IMPLEMENTATION AND SCREENSHOTS

6.1 IMPLEMENTATION

The Android application, Fridge Buddy is developed using many user defined activities which are based on developer-defined java classes. There are three Activities which are a part of the lifecycle of the application: MainActivity, BarcodeActivity and CaptureActivity.

MainActivity is the main view of the Fridge Buddy application which allows the user to add a product to the database. BarcodeActivity is called using an Intent. The authors of ZXing provide Intent Integrator classes to easily integrate barcode scanner into this project. CaptureActivity is also a sample activity provided by the author of Tess Two, which accesses the OCR module and provides a variable selectable region to scan the dates using OCR.

6.2 ADDING A NEW FEATURE

The way that the application has been developed, it makes it very easy to add new features to the application. So that the list can be customised by the user, a delete item feature has been implemented in the application.

The following are the steps that are followed to add this feature to the application:

i. Create a context menu xml file

A context menu xml file needs to be created so that context menu items can be added to the context menu which is shown when the user long clicks on an item in the list of items. The xml file contains the different menu items with their respective names. The xml file that was created is shown below.

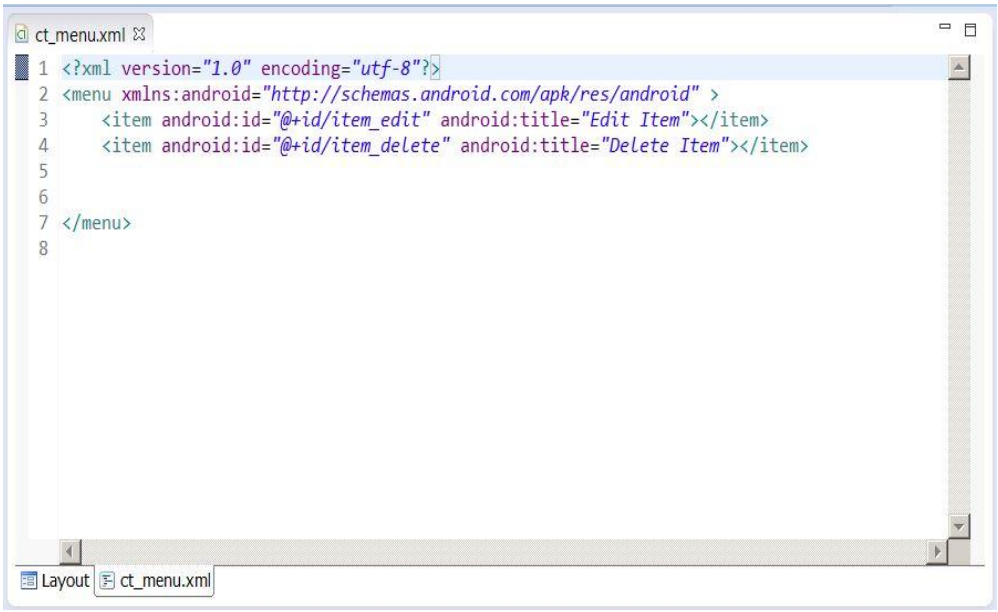

Figure 6.1 Context Menu XML file

ii. Add the onCreateContextMenu method to the ItemListView.java file

This method is added to the ItemListView activity so that the context menu xml file can be used to show the available options when the user long clicks on an item. The xml file is linked to this method by referring to it. The figure below shows the code that was added for this purpose.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
    // TODO Auto-generated method stub
    super.onCreateContextMenu(menu, v, menuInfo);
    getMenuInflater().inflate(R.menu.ct_menu, menu);
}
```

Figure 6.2 The onCreateContextMenu Method

iii. Add the onContextItemSelected method to the ItemListView,java file

This method is added to check which context menu item has been selected by the user and what should be done in each case. A delete menu item is added to the context menu which when selected will delete the item from the list and show the updated list. The complete code for adding the delete feature is shown in the figure below.

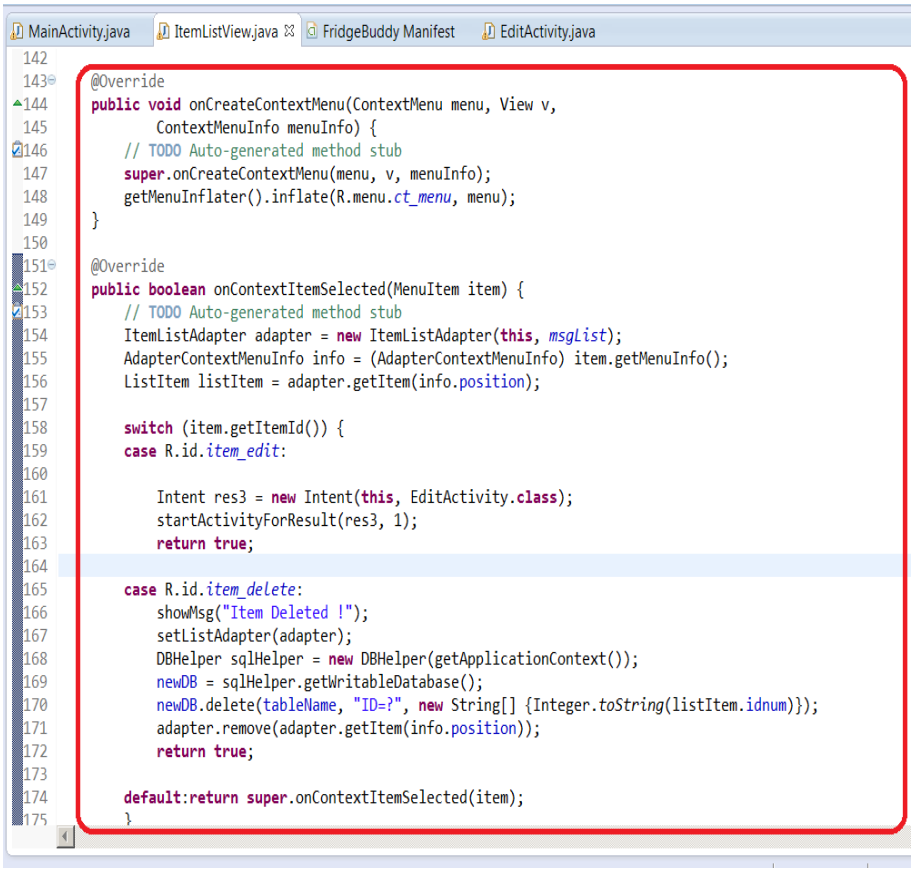

Figure 6.3 The code added for the delete feature

6.3 SCREENSHOTS

The screenshots of all the different parts of the application are shown below.

Figure 6.4 List Activity View when empty

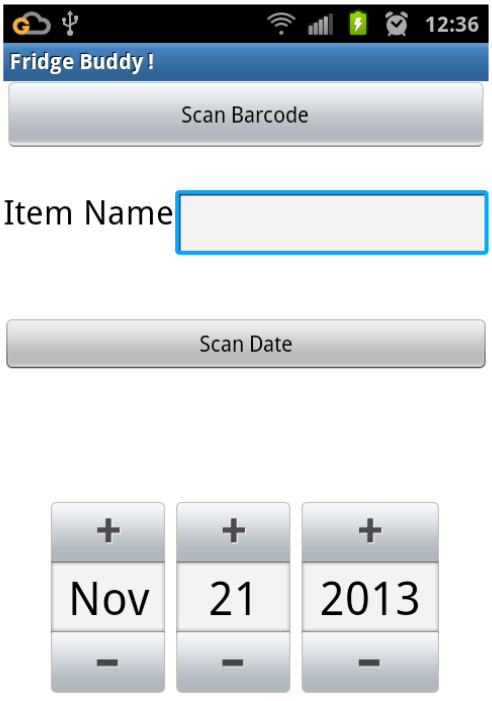

Figure 6.5 Main Activity View

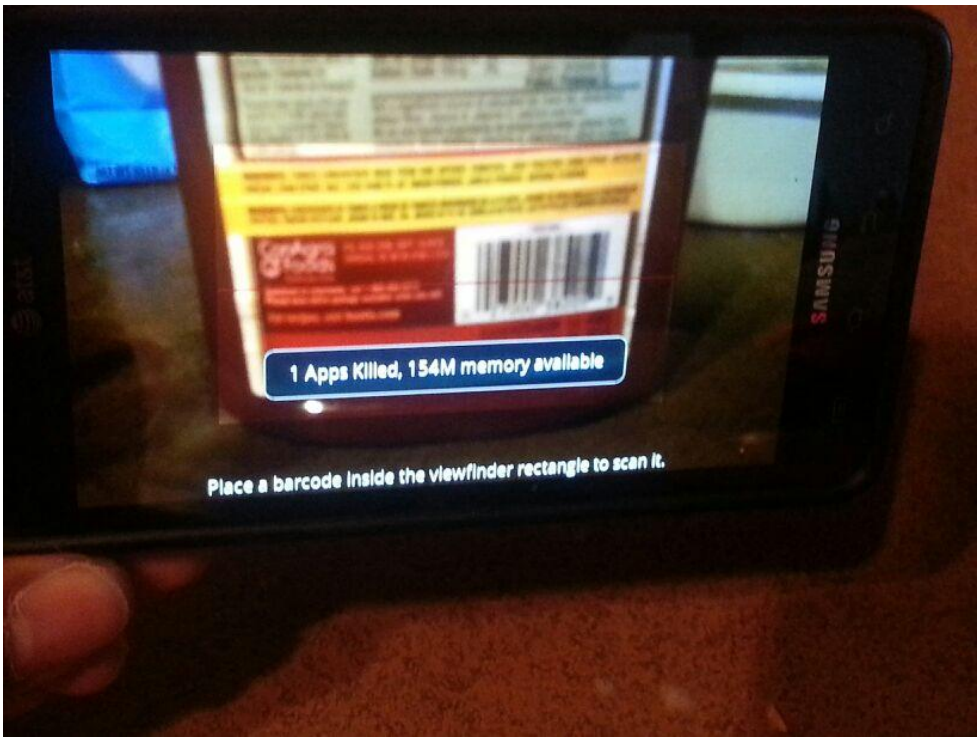

Figure 6.6 Barcode Scanning

Figure 6.7 Barcode Found

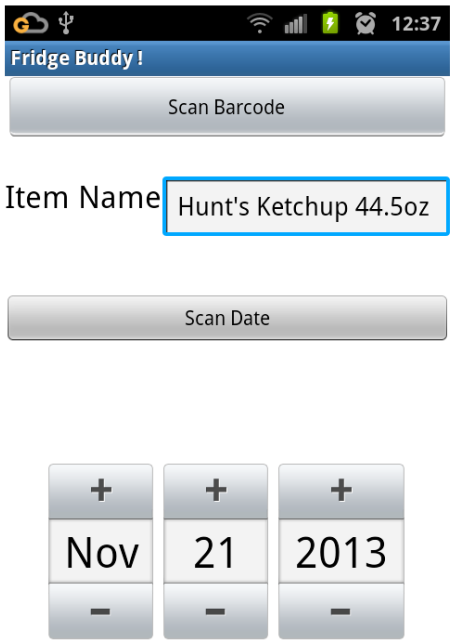

Figure 6.8 Main View with Item Name

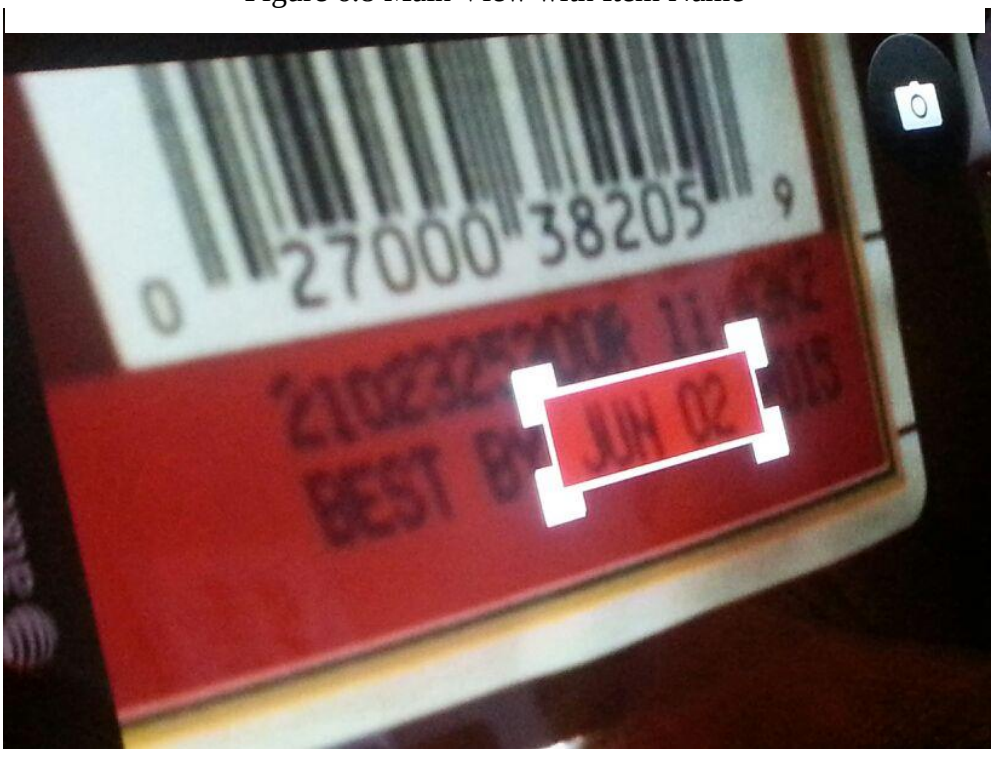

Figure 6.9 OCR Scanning

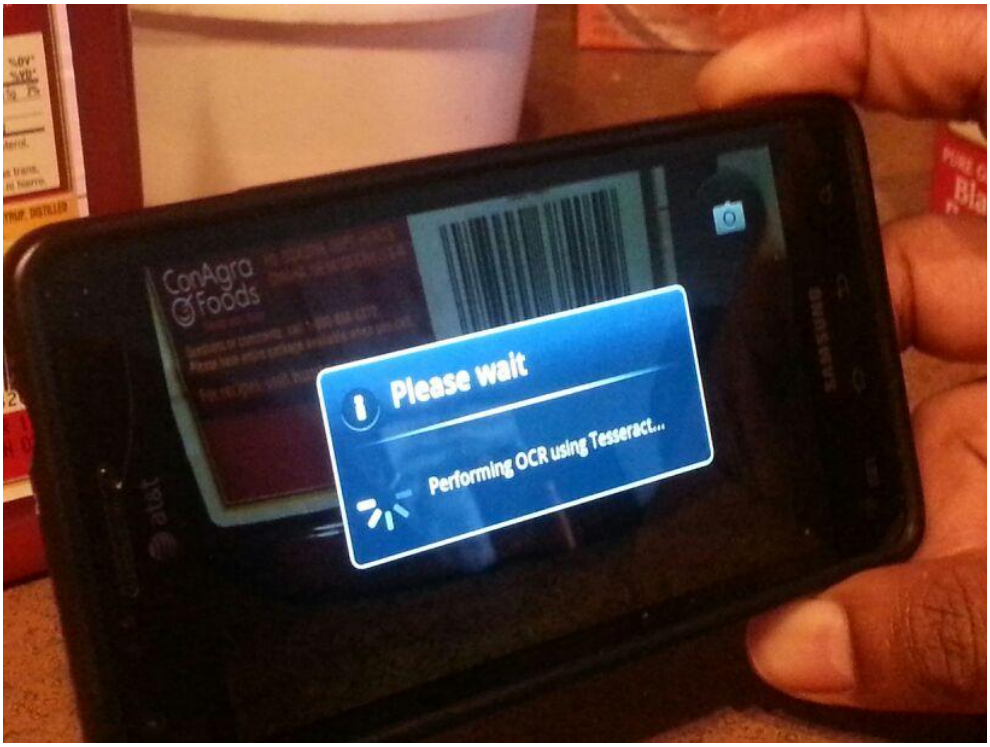

Figure 6.10 Performing OCR

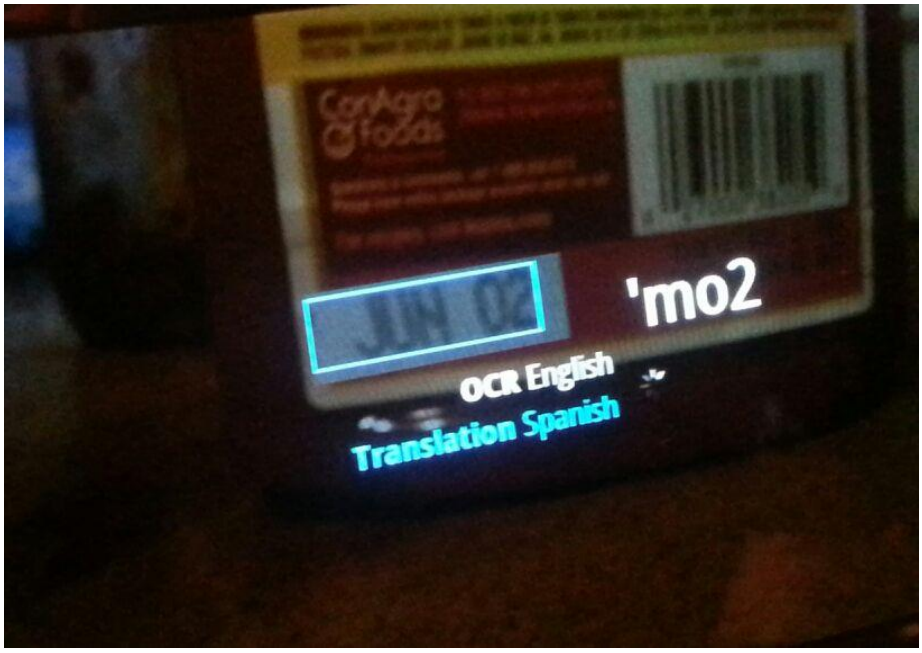

Figure 6.11 OCR Result

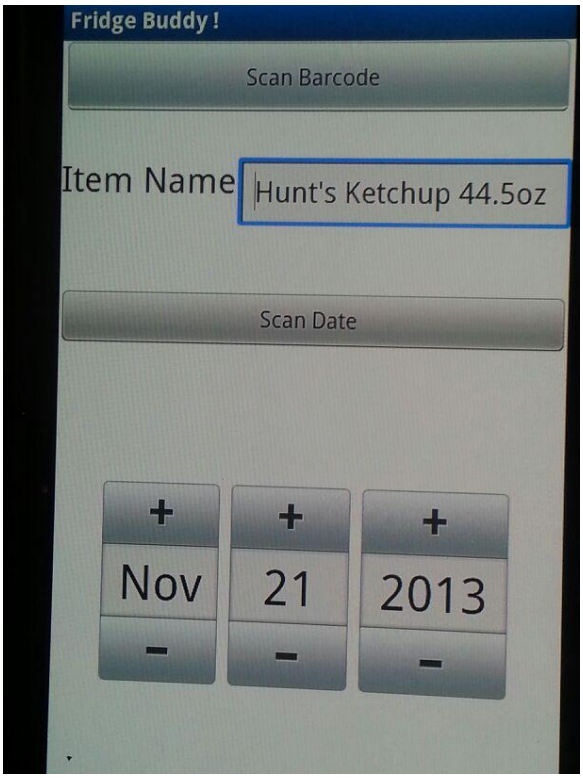

Figure 6.12 Main View after all information

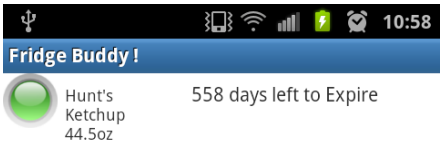

Figure 6.13 List View after adding Item

CHAPTER 7

TECHNOLOGIES USED

The technologies used to develop this application aims to reduce the time spent by the user to input the food item details and for implementing the different functionalities in the application. The technologies used are as follows :

i. Oracle Java 1.7 -

It is the principal programming language used for this application. It is an object oriented language which allows application developers to write code that runs on one platform but does not need to be recompiled on another. Java is also responsible for contributing libraries to parse JSON data retrieved from the Web Service. It also provides the SQLite libraries which enables the use of databases for the application.

ii. Google Android 2.2 Operating System -

This is minimum version of the supported mobile operating system on which this application can run on with all its features.

iii. Microsoft Windows Operating System -

This is the development platform which is being used to develop this application. It is easier to develop smart phone applications in Windows than in Linux as a lot of extra steps need to be implemented in Linux.

iv. SQLite:

This is the relational database engine which is used to store all the food item details. It is directly available as implementation classes in Java, as it is open source, and we can conveniently use it for our application. The preferences of the application are saved in

v. Eclipse IDE -

It is the integrated development environment that is being used to develop this application. It is an open source IDE which is readily available online and has a lot of useful features.

vi. ZXing -

This is the barcode scanning library that is used in this application to extract the barcodes from a food item. It is very easy to integrate in this project and is maintained by Google.

vii. Tess Two Library -

This is the optical character recognition library that is used to read the date of expiry and then manipulate it according to the requirements. This library is currently maintained by Google as well.

CHAPTER 8

FUTURE WORK AND CONCLUSION

8.1 FUTURE WORK

The future work that can be done are as follows :

i. Additional Training of OCR -

It is the principal programming language used for this application. It is an object oriented language which allows application developers to write code that runs on one platform but does not need to be recompiled on another. Java is also responsible for contributing libraries to parse JSON data

ii. Application for IPhone

This application can further be implemented in iOS to create an application for I Phone. It may be possible to use the same libraries for barcode scanning and OCR. The GUI would also be considerably for appealing in iOS.

iii. Better, more enhanced GUI

The user interface can be made more appealing to the user. More themes can be provided for the application to make the look and feel of the application more eye-catching.

iv. More features and preferences

The application can have more features like adding food items to different categories and different views like grid view or list view for displaying the food items. Additional preferences like turning off notifications globally or per item can also be added.

8.2 CONCLUSION

In conclusion, the application is able to show a list of all items by color coding it as well as sorting it in order of high priority. The application is also able to automatically discover the name of the products using barcode scanning and tries to detect the date of expiry by OCR scanning but is unsuccessful in doing so in most attempts. It was also able to provide preference editing options which makes the application easy to customize. Overall, this application would definitely be very useful in preventing food wastage as well as saving money.

REFERENCES

[1] Gábor Sörös, Christian Floerkemeier " Towards next generation barcode scanning", *Proc. Int. Conf. on Mobile and Ubiquitous Multimedia (MUM 2012), Ulm, Germany,* 2012

[2] R. Smith "An overview of the Tesseract OCR engine", *Proc. Int. Conf. Document Anal. Recognit.,* pp.629 -633 2007

[3] R. Adelmann "Mobile phone based interaction with everyday products - On the go", *Proc. IEEE Next Generation Mobile Applications, Services and Technologies, Cardiff, UK,* 2007

[4] Gunders, Dana. "Wasted: How America Is Losing Up to 40 Percent of Its Food from Farm to Fork to Landfill". *IP:12-06-B. New York, NY: National Resources Defense Council,*2012

[5] "Best Before" - https://play.google.com/store/apps/details?id=com.nicedistractions.bestbefore&hl=en

[6] "Fridge Police" - https://itunes.apple.com/us/app/fridge-police-food-expiratio/id394119420?mt=8&ign-mpt=uo%3D4

[7] "Food Expiration Track" - https://play.google.com/store/apps/details?id=com.touchsi.keepemfresh

[8] "Food Expiration Saver"- https://play.google.com/store/apps/details?id=tw.com.expiration

[9] "Shelf Life" - https://itunes.apple.com/us/app/shelf-life-food-expiration/id292997652?mt=8

APPENDIX A – CODE

1. ItemListView.java

```
package com.FridgeBuddy.barcodescanning;

import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import android.app.ListActivity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.ParseException;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.util.Log;
import android.view.ContextMenu;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.AdapterView.AdapterContextMenuInfo;
import android.widget.Toast;

public class ItemListView extends ListActivity {
        private static final int INSERT_ID = Menu.FIRST;
```

```
        private static final int SETTINGS_ID = Menu.FIRST+1;
        private static final int QUIT_ID = Menu.NONE;
        private Cursor mNotesCursor;
        private String tableName = DBHelper.tableName;
        private static ArrayList<ListItem> msgList = new ArrayList<ListItem>();
        private Cursor c;
        SQLiteDatabase newDB;
        /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.notes_list);

      fillData();
      registerForContextMenu(getListView());
    }

    @SuppressWarnings("deprecation")
        private void fillData() {


        try {
                msgList.clear();
                DBHelper sqlHelper = new DBHelper(getApplicationContext());
                newDB = sqlHelper.getWritableDatabase();
                c = newDB.rawQuery("SELECT ID, ITEMNAME,
DATEOFEXPIRY FROM " + tableName + " ORDER BY DATEOFEXPIRY ASC",
null);

        if (c != null ) {
                if  (c.moveToFirst()) {
                        do {
                                int id = c.getInt(c.getColumnIndex("ID"));
                                String itemName =
c.getString(c.getColumnIndex("ITEMNAME"));
                                String dateOfExp =
c.getString(c.getColumnIndex("DATEOFEXPIRY"));



                                        Calendar c = Calendar.getInstance();

                                        SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
                                    Date currentDate = (Date) c.getTime();
                                        Date convertedDate = new Date();
```

```
                                        try {
                                           convertedDate =
dateFormat.parse(dateOfExp);
                                        } catch (ParseException e) {
                                           // TODO Auto-generated catch block
                                           e.printStackTrace();
                                        } catch (java.text.ParseException e) {
                                                                // TODO Auto-
generated catch block
                                                                e.printStackTrace();
                                                        }


                                        String NoDays =
getDateDiffString(convertedDate,currentDate);
                                        msgList.add(new ListItem(itemName,
NoDays,id));


                                }while (c.moveToNext());
                        }
                }
                c.close();
                newDB.close();


                }
                catch (SQLiteException se ) {
                Log.e(getClass().getSimpleName(), "Could not open the cursor");
        }
                finally {
                        newDB.close();
        }

                ItemListAdapter adapter = new ItemListAdapter(this, msgList);
                setListAdapter(adapter);

   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);
      menu.add(0, INSERT_ID, 0, R.string.menu_insert);
      menu.add(0, SETTINGS_ID, 0, R.string.menu_preferences);
      menu.add(0, QUIT_ID, 0, R.string.menu_quit);
      return true;
```

```
}

@Override
public boolean onMenuItemSelected(int featureId, MenuItem item) {
   switch(item.getItemId()) {
      case INSERT_ID:
            Intent res = new Intent(this, MainActivity.class);
                  startActivityForResult(res, 1);
         return true;

      case SETTINGS_ID:
            Intent res2 = new Intent(this, Prefs.class);
                  startActivityForResult(res2, 1);
         return true;
      case QUIT_ID:
            finish();
         return true;
   }

   return super.onMenuItemSelected(featureId, item);
}

@Override
      public void onCreateContextMenu(ContextMenu menu, View v,
                  ContextMenuInfo menuInfo) {
            // TODO Auto-generated method stub
            super.onCreateContextMenu(menu, v, menuInfo);
            getMenuInflater().inflate(R.menu.ct_menu, menu);
      }

@Override
      public boolean onContextItemSelected(MenuItem item) {
            // TODO Auto-generated method stub
      ItemListAdapter adapter = new ItemListAdapter(this, msgList);
      AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
      ListItem listItem = adapter.getItem(info.position);

      switch (item.getItemId()) {
      case R.id.item_edit:

            Intent res3 = new Intent(this, EditActivity.class);
                  startActivityForResult(res3, 1);
            return true;

      case R.id.item_delete:
            showMsg("Item Deleted !");
```

```
                setListAdapter(adapter);
                DBHelper sqlHelper = new DBHelper(getApplicationContext());
                        newDB = sqlHelper.getWritableDatabase();
                        newDB.delete(tableName, "ID=?", new String[]
{Integer.toString(listItem.idnum)});
                        adapter.remove(adapter.getItem(info.position));
                return true;

        default:return super.onContextItemSelected(item);
        }

        }


   protected void onActivityResult(int requestCode, int resultCode, Intent data) {

         if (requestCode == 1) {

           if(resultCode == RESULT_OK){
              String itemName=data.getStringExtra("item");
              String dateExp=data.getStringExtra("date");
              DBHelper sqlHelper = new DBHelper(getApplicationContext());
              newDB = sqlHelper.getWritableDatabase();


              sqlHelper.insertIntoDatabase(itemName, dateExp);
              newDB.close();
              fillData();
           }
           if (resultCode == RESULT_CANCELED) {
              //Write your code if there's no result
           }
         }
        }//o

   @Override
   protected void onDestroy() {
        super.onDestroy();

   }

   @Override
   protected void onResume() {
        super.onResume();
        //mDbHelper = new DBHelper(this);
        fillData();
```

```
}

@Override
protected void onPause() {
	super.onPause();
	//mDbHelper.close();
}
class ListItem{
	public ListItem(String itemName2, String dateOfExp, int id) {
			itemName = itemName2;
			date = dateOfExp;
			idnum=id;
		}
		String itemName;
	String date;
	int idnum;
}




	/**
	 *  Returns a string that describes the number of days
	 *  between dateOne and dateTwo.
	 *
	 */

public String getDateDiffString(Date dateOne, Date dateTwo)
	{
	   long timeOne = dateOne.getTime();
	   long timeTwo = dateTwo.getTime();
	   long oneDay = 1000 * 60 * 60 * 24;
	   timeOne+=oneDay;
	   long delta = (timeTwo - timeOne) / oneDay;

	  if (delta >= 0 ) {
		if(delta==1)
			return "EXPIRED !! " + delta + " day ";
	    else
			return "EXPIRED !! " + delta + " days ";
	  }
	   else {
	     delta *= -1;
	     if(delta==1)
```

```
                return "" + delta + " day left to Expire";
            else
                return "" + delta + " days left to Expire";
        }
    }

  private void showMsg(String message) {
                Toast msg = Toast.makeText(ItemListView.this, message,
Toast.LENGTH_LONG);
                msg.setGravity(Gravity.CENTER, msg.getXOffset() / 2,
                                msg.getYOffset() / 2);
                msg.show();
    }

}



    }
```

2. MainActivity.java

```
package com.FridgeBuddy.barcodescanning;

import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import org.json.JSONException;
import org.json.JSONObject;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.os.SystemClock;
import android.preference.PreferenceManager;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
```

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.support.v4.app.NotificationCompat;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.view.Menu;
import com.FridgeBuddy.barcodescanning.ocr.CaptureActivity;
import com.FridgeBuddy.jsonparsing.JSONParser;
import com.FridgeBuddy.barcodescanning.R;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;
//import com.FridgeBuddy.barcodescanning.DBAdapter;;

public class MainActivity extends Activity implements OnClickListener {

        private Button scanBtn;
        private Button ocrBtn;
        private Button submitBtn;
        private TextView formatTxt, contentTxt,validTxt,numberTxt;
        private EditText itemnameTxt;
        private TextView itemlabelTxt;

        //newly added
   public static final String NOTIFICATION_DATA = "NOTIFICATION_DATA";

        private DatePicker date1;
        // url to make request
                private static String url =
"http://upcdatabase.org/api/json/a65fbd5cec11e92233ddbb55f00c8921/";

                // JSON Node names
                private static final String TAG_VALID = "valid";
                private static final String TAG_NUMBER = "number";
                private static final String TAG_ITEMNAME = "itemname";
                private static final String TAG_DESCRIPTION = "description";
                private static final String TAG_PRICE = "price";
                private static final String TAG_RATINGSUP = "ratingsup";
                private static final String TAG_RATINGSDOWN = "ratingsdown";

                //DB
        //      DBAdapter db = new DBAdapter(this);
```

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		scanBtn = (Button)findViewById(R.id.scan_button);
		ocrBtn = (Button)findViewById(R.id.ocr_button);
		submitBtn = (Button)findViewById(R.id.submit_button);
		itemnameTxt = (EditText)findViewById(R.id.itemname);
		scanBtn.setOnClickListener(this);
		ocrBtn.setOnClickListener(this);
		submitBtn.setOnClickListener(this);
		date1 = (DatePicker)findViewById(R.id.datePicker1);

	}


	//NOTIFICATION SERVICE CODE
	@Override
	public void onResume() {
	   super.onResume();
	   SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
	   int minutes = prefs.getInt("interval",0);
	   AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
	   Intent i = new Intent(this, NotificationService.class);
	   PendingIntent pi = PendingIntent.getService(this, 0, i, 0);
	   am.cancel(pi);
	   // by my own convention, minutes <= 0 means notifications are disabled
	   if (minutes > 0) {
	      am.setInexactRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
	         SystemClock.elapsedRealtime() + minutes*60*1000,
	         minutes*60*1000, pi);
	   }
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}

	@SuppressWarnings("deprecation")
	public void onClick(View v){
		//respond to clicks
		if(v.getId()==R.id.scan_button){
```

```
                    //scan
                    IntentIntegrator scanIntegrator = new IntentIntegrator(this);
                    scanIntegrator.initiateScan();
                    }
                if(v.getId()==R.id.ocr_button){
                    //scan OCR
                Calendar cal = Calendar.getInstance();

                    int year = cal.get(Calendar.YEAR);
            int month = cal.get(Calendar.MONTH);
            int day = cal.get(Calendar.DAY_OF_MONTH);

           // cal.set(year, month + 1, day);
               //      cal.set(date1.getYear(), date1.getMonth() + 1,
date1.getDayOfMonth());
                    cal.add(Calendar.DATE, 7);
                    date1.updateDate(year, month-1,
cal.get(Calendar.DAY_OF_MONTH));



                    Intent res = new Intent(this, CaptureActivity.class);
                    startActivity(res);

                    }
                if(v.getId()==R.id.submit_button){
                    //Submit. Add Item to database

                    Calendar cal = Calendar.getInstance();

                    int year = cal.get(Calendar.YEAR);
            int month = cal.get(Calendar.MONTH);
            int day = cal.get(Calendar.DAY_OF_MONTH);

            cal.set(year, month + 1, day);

            Intent returnIntent = new Intent();
                     returnIntent.putExtra("item",itemnameTxt.getText().toString());
                     Date date2 = new Date(date1.getYear()-1900, date1.getMonth(),
date1.getDayOfMonth());
                     //Date date = (Date) new SimpleDateFormat("yyyy-MM-
dd").parse(date2.toString());
                     returnIntent.putExtra("date",date2.toString());
                     setResult(RESULT_OK,returnIntent);
```

```
                int hour = 10;
                int minute = 10;

                    Calendar calendar = Calendar.getInstance(); // CALENDAR
                    calendar.set(date2.getYear(), date2.getMonth(), date2.getDate(),
                            hour, minute, 0);
                    long startTime = calendar.getTimeInMillis()+120000;
                    //NOTIFICATIONS CODE
                            createNotification(System.currentTimeMillis(),
itemnameTxt.getText().toString());
                    //createNotification(startTime, itemnameTxt.getText().toString());


                    finish();

                   }

            }

        private void createNotification(long when,String data){
                String notificationContent ="Click Here to see your list ";
                String notificationTitle ="Your "+data + " is about to expire !!";
                //large icon for notification,normally use App icon
                Bitmap largeIcon =
BitmapFactory.decodeResource(getResources(),R.drawable.ic_launcher);
                int smalIcon =R.drawable.ic_launcher;
                String notificationData="This is data : "+data;

                /*create intent for show notification details when user clicks notification*/
                Intent intent =new Intent(getApplicationContext(), ItemListView.class);
                intent.putExtra(NOTIFICATION_DATA, notificationData);

                /*create unique this intent from  other intent using setData */
                intent.setData(Uri.parse("content://"+when));
                /*create new task for each notification with pending intent so we set
Intent.FLAG_ACTIVITY_NEW_TASK */
                PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(), 0, intent,
Intent.FLAG_ACTIVITY_NEW_TASK);

                /*get the system service that manage notification NotificationManager*/
                NotificationManager notificationManager =(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);

                /*build the notification*/
```

```
            NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(
                                    getApplicationContext())
                                    .setWhen(when)
                                    .setContentText(notificationContent)
                                    .setContentTitle(notificationTitle)
                                    .setSmallIcon(smalIcon)
                                    .setAutoCancel(true)
                                    .setTicker(notificationTitle)
                                    .setLargeIcon(largeIcon)
                                    .setDefaults(Notification.DEFAULT_LIGHTS|
Notification.DEFAULT_VIBRATE| Notification.DEFAULT_SOUND)
                                    .setContentIntent(pendingIntent);

            /*Create notification with builder*/
            Notification notification=notificationBuilder.build();

            /*sending notification to system.Here we use unique id (when)for making
different each notification
             * if we use same id,then first notification replace by the last notification*/
            notificationManager.notify((int) when, notification);
        }

        public void onActivityResult(int requestCode, int resultCode, Intent intent) {
            //retrieve scan result
            IntentResult scanningResult =
IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);

            if (scanningResult != null) {
                //we have a result
                String scanContent = scanningResult.getContents();
                String scanFormat = scanningResult.getFormatName();

                getDetailsonline(scanContent);


            }
            else{
               Toast toast = Toast.makeText(getApplicationContext(),
                  "No scan data received!", Toast.LENGTH_SHORT);
               toast.show();

            }
        }

        private void getDetailsonline(String scanContent) {
```

```
                // Hashmap for ListView
                                ArrayList<HashMap<String, String>> contactList = new
ArrayList<HashMap<String, String>>();


                                if(haveNetworkConnection())
                                {
                                // Creating JSON Parser instance

                                JSONParser jParser = new JSONParser();

                                // getting JSON string from URL
                                JSONObject json =
jParser.getJSONFromUrl(url+scanContent);

                                try {
                                                // Storing each json item in variable
                                                String valid = json.getString(TAG_VALID);
                                                String number =
json.getString(TAG_NUMBER);
                                                String itemname =
json.getString(TAG_ITEMNAME);
                                                String description =
json.getString(TAG_DESCRIPTION);
                                                String price = json.getString(TAG_PRICE);
                                                String ratingsup =
json.getString(TAG_RATINGSUP);
                                                String ratingsdown =
json.getString(TAG_RATINGSDOWN);


                                                itemnameTxt.setText(itemname);


                                } catch (JSONException e) {
                                        e.printStackTrace();
                                }

                // TODO Auto-generated method stub

        }
}

        private boolean haveNetworkConnection() {
                // TODO Auto-generated method stub
                 boolean haveConnectedWifi = false;
```

```
                boolean haveConnectedMobile = false;

                ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
                NetworkInfo[] netInfo = cm.getAllNetworkInfo();
                for (NetworkInfo ni : netInfo)
                {
                   if (ni.getTypeName().equalsIgnoreCase("WIFI"))
                      if (ni.isConnected())
                         haveConnectedWifi = true;
                   if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
                      if (ni.isConnected())
                         haveConnectedMobile = true;
                }
                return haveConnectedWifi || haveConnectedMobile;
        }




        }
```

3. JSONParser.java

```
package net.niceandroid.jsonparsing;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;
import android.util.Log;

public class JSONParser {

        static InputStream is = null;
        static JSONObject jObj = null;
        static String json = "";
```

```
// constructor
public JSONParser() {

}

public JSONObject getJSONFromUrl(String url) {

        // Making HTTP request
        try {
                // defaultHttpClient
                DefaultHttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);

                HttpResponse httpResponse = httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }

        try {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(
                                is, "iso-8859-1"), 8);
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                }
                is.close();
                json = sb.toString();
        } catch (Exception e) {
                Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try parse the string to a JSON object
        try {
                jObj = new JSONObject(json);
        } catch (JSONException e) {
                Log.e("JSON Parser", "Error parsing data " + e.toString());
        }
```

```
                // return JSON String
                return jObj;

        }
}
```

4. DBHelper.java

```
package com.FridgeBuddy.barcodescanning;

import java.sql.Date;

import android.content.ContentValues;
import android.content.Context;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBHelper extends SQLiteOpenHelper {

        public static SQLiteDatabase DB;
        public String DBPath;
        public static String DBName = "FridgeBuddyDb3";
        public static final int version = '1';
        public static Context currentContext;
        public static String tableName = "Items";

        public DBHelper(Context context) {
                super(context, DBName, null, version);
                currentContext = context;
//              DBPath = "/data/data/" + context.getPackageName() + "/databases/";
                DBPath = "/mnt/";
                Log.d("DBPAth", "DBPath is " + DBPath);
//              clearDatabase();
                createDatabase();
        }

        @Override
        public void onCreate(SQLiteDatabase arg0) {
                // TODO Auto-generated method stub

        }

        @Override
```

```
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		// TODO Auto-generated method stub

	}

	private void createDatabase() {
		boolean dbExists = checkDbExists();

		if (dbExists) {
			Log.d("DBHELPER", "DB already exists");
		} else {
			DB = currentContext.openOrCreateDatabase(DBName, 0, null);
			DB.execSQL("CREATE TABLE IF NOT EXISTS " + tableName
+
			" (ID integer primary key autoincrement, "
	+ "ITEMNAME VARCHAR not null, DATEOFEXPIRY datetime);");
			Log.d("DBHELPER", "Table creation check");
		}
	}

	private boolean checkDbExists() {
		SQLiteDatabase checkDB = null;
		try {
			String myPath = DBPath + DBName;
			checkDB = SQLiteDatabase.openDatabase(myPath, null,
SQLiteDatabase.OPEN_READONLY);

		} catch (SQLException e) {
			e.printStackTrace();
			Log.e(getClass().getSimpleName(), "Could not open the
database");
		}

		if (checkDB != null) {
			Log.d("DBHELPER","DB exists");
			checkDB.close();
		}

		return checkDB != null ? true : false;
	}

	public static boolean insertIntoDatabase(String itemName, String dateExp){
		ContentValues contentValues = new ContentValues();
		contentValues.put("ITEMNAME", itemName);
		contentValues.put("DATEOFEXPIRY", dateExp);
```

```
                DB = currentContext.openOrCreateDatabase(DBName, 0, null);
                if (DB.insert(tableName, null, contentValues)<0){
                        Log.d("DBHELPER", "Insert Failed");
                        return false;
                }
                else{
                        Log.d("DBHELPER", "Insert Successful");
                        return true;
                }
        }

        public static void clearDatabase(){
                DB = currentContext.openOrCreateDatabase(DBName, 0, null);
                DB.execSQL("DELETE FROM " + tableName + ";");
                Log.d("DBHELPER", "Table cleared.");
        }
}
```

5. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.FridgeBuddy.barcodescanning"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.CAMERA"/>
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
 <uses-feature android:name="android.hardware.camera.autofocus" />
  <uses-feature android:name="android.hardware.camera.flash" android:required="false"
/>
  <uses-feature android:name="android.hardware.camera"/>
  <uses-feature android:name="android.hardware.screen.landscape"/>
   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
```

```
        android:theme="@style/AppTheme"
        android:debuggable="true" >

        <activity
            android:name="com.FridgeBuddy.barcodescanning.MainActivity"
            android:label="@string/app_name" >
                </activity>

        <activity
            android:name="com.FridgeBuddy.barcodescanning.ItemListView"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.FridgeBuddy.barcodescanning.ocr.CaptureActivity"
              android:screenOrientation="landscape"
              android:configChanges="orientation|keyboardHidden|screenSize"
              android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
              android:windowSoftInputMode="stateAlwaysHidden"
              >

            </activity>
            <activity android:name="com.FridgeBuddy.barcodescanning.ocr.HelpActivity"
              android:screenOrientation="user">
      <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>
    <activity
            android:name="com.FridgeBuddy.barcodescanning.EditActivity"
             />

    <activity android:name="com.FridgeBuddy.barcodescanning.ocr.PreferencesActivity"
/>
    <activity android:name=".Prefs" android:theme="@android:style/Theme.Light"/>
    <receiver android:name=".BootReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>

    </application>
```

```
</manifest>
```

6. activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

   <Button
      android:id="@+id/scan_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"

      android:layout_alignParentRight="true"
      android:text="@string/scan" />

   <EditText
      android:id="@+id/itemname"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_below="@+id/scan_button"
      android:layout_marginTop="23dp"
      android:ems="10"
       />

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/itemname"
      android:layout_alignParentLeft="true"
      android:layout_alignTop="@+id/itemname"
      android:text="Item Name"
      android:textAppearance="?android:attr/textAppearanceLarge" />

   <Button
      android:id="@+id/ocr_button"

      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentRight="true"
      android:layout_below="@+id/itemname"
```

```
        android:layout_marginTop="38dp"
        android:minWidth="64dip"
        android:text="@string/scan1" />

    <Button
        android:id="@+id/submit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="Submit" />

    <DatePicker
        android:id="@+id/datePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_below="@+id/ocr_button" />

</RelativeLayout>
```