PROJEKTIN LOPPURAPORTTI - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

FOOD SCANNER

Barcode reader

TEKIJÄ/T: Shulin Yao

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietojenkäsittelyn koulutusohjelma	
Työn tekijä(t) Shulin Yao	
Työn nimi Food Scanner	
Päiväys 01.05.2015	Sivumäärä/Liitteet 36
Ohjaaja(t) Mr Jussi Koistinen, Lecturer and Mr Keijo Kuosmanen, Lecturer	
Toimeksiantaja/Yhteistyökumppani(t) Mr Arto Toppinen	

Tiivistelmä

Nykyään kädessä pidettävät laitteet, kuten tabletit ja älypuhelimet, kehittyvät ja niiden määrä lisääntyy. Tällaiset laitteet, jotka sisältävät useita lisäominaisuuksia, ovat avanneet oven monenlaisille kaupallisille mahdollisuuksille. Nykyään älypuhelimet yleensä sisältävät jo kameran, Internet-yhteyden ja prosessorin. Älypuhelimet voivat tarjota joitakin samoja palveluja kuin tietokone. Sisällä toimivalla sovelluksella on mahdollista suorittaa päivittäisiä toimintoja, kuten kuunnella musiikkia ja katsella videoita verkossa.

Tässä työssä kehitettiin ruokatuotteiden viivakoodin lukija Android-alustalle. Opinnäytetyössä harjoiteltiin Android sovelluksen kehittämistä, opiskeltiin verkkosuunnittelua, integroitiin ulkoinen kirjasto Android-projektiin ja tutkittiin miten ne toimivat asiakkaan ja palvelimen välillä.

Tämän projektin aikana käytettiin jäljempänä kuvattuja menetelmiä. Ensiksi luotiin Android-sovellus käyttäen Android studiota.. Toiseksi rakennettiin paikallinen tietokanta palvelimelle käyttämällä XAMPP:ia. Kolmanneksi rakennettiin verkkopalvelu sovelluksen ja eri toimialueella sijaitsevan verkkopalvelimen välille. HTTPCLIENT:ia käytettiin yhteyksien muodostamiseen ja kommunikaatioon. Lopuksi Json:ia käytettiin tietojen vaihdon objektina sovelluksen, asiakkaan ja verkkopalvelimen välillä

Lopulta, Työssä saatiin paljon uutta tietoa kommunikaatiosta sovelluksen ja palvelimen välillä. Sovelluksen testauksen jälkeen, sen voi ajaa menestyksekkäästi. Se voi skannata viivakoodin hyvin, saada sisältöä ja kirjoitaa viivakoodeja. sen voi myös yhdistää palvelimeen ja vastaanottaa elintarvikeiden tietoa FoodAurora

Avainsanat
Puhelin, android-sovellus, viivakoodi, ZXING, JSON, HTTPCLIENT

SAVONIA UNIVERSITY OF APPLIED SCIENCES — THESIS Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Shulin Yao			
Title of Thesis Food Scanner			
Date	01 May 2015	Pages/Appendices	36
Supervisor(s) Mr Jussi Koistinen, Lecturer and Mr Keijo Kuosmanen, Lecturer			
Client Organisation /Partners Mr Arto Toppinen			

Abstract

Today hand-held devices such as tablets and smart phones are developing and increasing. These kinds of devices which include a number of extra features have opened a door to a wide range of commercial possibilities. Nowadays, normally smart phones already include a camera, the internet access and a processor like PCs include. Smart phones can offer some services like PCs. With more applications working inside, it is possible to use them to accomplish daily sessions, such as listening to music and watching videos online.

In this case, there was one android application that fell into this category, a barcode scanner for food. The purpose of this thesis was to practice android development, to learn web development, to integrate the external library into an android project and to understand how they worked between the client and server.

During the project, several methods were used. Firstly, based on the development platform of the android studio, android programming was used. Secondly, a local database server with XAMPP was used. Thirdly, a web service was built between the application and the web server which was not in the same domain. Thus, HTTPCLIENT was used for connection and communication. Finally, JSON was used as the data exchange object between the application client and the web server.

Finally, a lot of information such as scanning barcode and receiving food information from foodaurora was gained about communication between the application and the server. After testing the application, it can run successfully and scan barcode well, get contents and type on barcodes. It can also access server and receive food information from FoodAurora.

Keywords
smart phone, android application, barcode, ZXING, JSON, HTTPCLIENT

Contents

1 INTRODUCTION

1.1 Motivation

Nowadays, Smartphones are sold and used very widely. At the same time, the related products and services are offered. So smartphones have many functionalities, they can work with a variety of sensors, installable applications and have the Internet access. And it also can get interaction with the Internet of Things (IOT). IoT means some different information sensing equipment, such as radio frequency identification devices, infrared sensors, global positioning systems, laser scanners and various other devices and the Internet combine to bullid a large network. The aim is to have all the items are linked to the network, to facilitate the identification and management, to offer service and get much value by exchanging data with the manufacturer and some connected devices. Based on the embeded computing system, each thing can be identified uniquely. (Wikipedia IOT 2015.) So in this field, these related industries scenarios get a potential for a large number of consumer-oriented services which can trigger a new service market for the IOT. Meanwhile, more and more consumers use smartphones, and their Internet life will not only on a web, but also most on a mobile, not only keeping their activities on the internet. However, so lots of products and services are offered, so how to identify single products uniquely. This is done by the global trade item number (GTIN) which is an identifier for trade items developed by comprising among others of the former EAN International and Uniform Code Council (GS1), this kind od identifiers are used for searching product information in a database which may be held by the manufacturer, some retailer and other entity. The European article number (EAN) also has same kind of functionality to offer to support this service which is used to Distinguish different products which even has same package, but different amounts inside. (Wikipedia GTIN 2015.)

1.2 Project Description

The functionalities can be implemented after completing the whole project as follows:

- Firstly, it needs to develop the basic structure of the android application based on the android studio. It should include three basic activities to display, which is to show interface, camera scanning and product details.
- Secondly, Zebra Crossing which is an open source library to trigger camera and handle the barcode is added to read barcode information.
- Thirdly, it can be used to connect the web to access an online database by Api offered

Thus, the purpose of the whole project is to develop an android application that is consumer-oriented. This application would offer the service as a personal shopping assistant which allows consumers to scan product information from an external database, they can send and share information on the social network or e-mail.

2 BARCODE

2.1 Barcode types

Nowadays, there are many different barcode types that exist for many different purposes. Normally, they can be divided into 1D and 2D barcodes.

The reason why so many different types of barcodes exist is that most of them need to use in a wide range of operational areas. Thus, it is important and also efficient to select the most suitable barcodes to meet the requirements of a special field.

2.1.1 Linear 1D Barcodes

There are many different types of Linear Barcodes as Figure 1 shows. It encodes the information in horizontally, so they are also called one-dimensional barcodes (1D). Most people may think barcodes are columns of varying width lines that are attached on the back of products. It includes EAN-13/UPC-A, Code 128, Code 39, EAN-8 etc. As more data is encoded, 1D barcodes gets longer, and they grows, this is also the reason why the space of 1D is inefficient. (Omniplanar BarcodesDefinition 2015.)

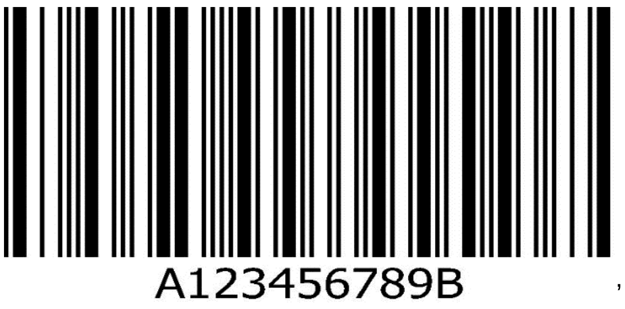


FIGURE 1. One kind of barcode which is a CODABAR code

2.1.2 2D Barcodes (Stacked)

PDF417 is one of the most popular 2D codes because of its ability to be read with slightly modified handheld laser or linear CCD scanners. CCD means a charge-coupled device (CCD) which is a semiconductor device, capable of the optical image into a digital signal. Tiny implanted on the CCD photosensitive substance called pixels (Pixel). The more pixels on a CCD contains, the more it provides the screen resolution is higher. Like film, but it convert an optical signal into an electric charge signal. There are many rows of photodiodes which can sense light, and can convert an optical signal into an electrical signal, after amplification and sampling by the external analog-digital conversion circuit is converted into a digital image signal. (Wikipedia CCD 2015.). It is 2-dimensional stacked barcode created by Symbol Technologies in 1991. PDF stands for Portable Data File and 417 means the 17 modules of 4 bars and spaces that make up each code. Each symbol also has started and stopped bar groups that allow the code to be easily identified as Figure 2 shows. (Omniplanar-Barcodes Definition 2015.)

Two-dimensional barcodes are known under names like PDF417, or Codablock F

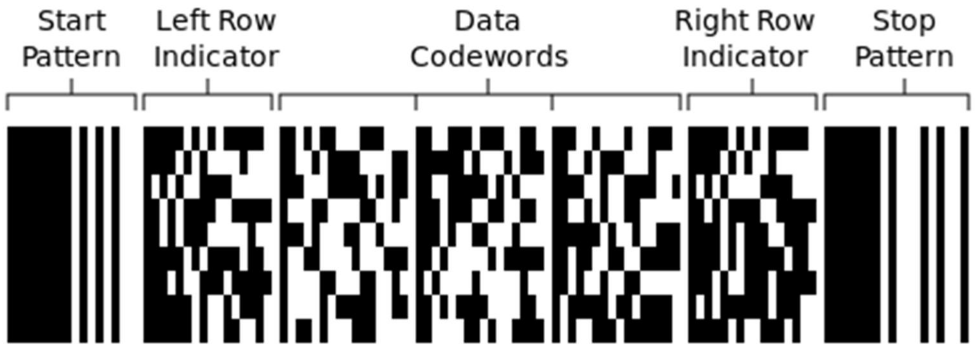


FIGURE 2. A 2D Stacked Barcode PDF 417 sample

2.1.3 2D Barcodes (Matrix Codes)

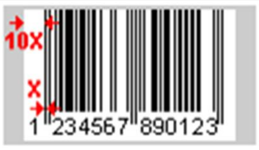
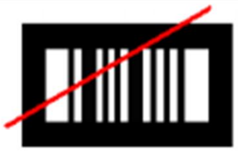
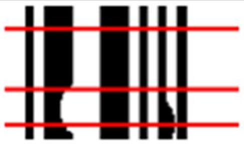
A Data Matrix code as figure 3 shows is a two-dimensional matrix barcode which consists of black and white square modules arranged in either a square or rectangular pattern. It is a high-density, two-dimensional (2D) symbology that encodes text, numbers, files and actual data bytes, also including Unicode characters and photos. The length of the encoded data depends on the symbol dimension used. Error correction codes are added to increase symbol strength: even if they are partially damaged, they can still be read. (Omniplanar-Barcodes Definition, 2015)

FIGURE 3. One kind of a 2D barcode (QR)

2.2 Barcode Glossary

The following Table 1 shows explanation of the terms which are used in a barcode. (TEC-IT, 11 2015.)

TABLE 1. Barcode Glossary

Bar	A bar is represented by the dark or black elements in a barcode.	1 23456789012 3
Space	The white or lighter elements in a barcode are called spaces.	
Barcode density	The density of the barcode refers to how much space is required for the needed characters (characters per Inch or centimeter)	
Element	Represents both a bar and a space.	
Module	A module is the smallest element of a barcode. The width of the single bars and spaces is a (mostly integer) multiples of the basic width of the module.	
Module width	The width of the barcode's smallest element in millimeter, in inches or in so-called mils (one mil = 1/1000 inch). The module width is usually abbreviated with the letter X.	
X Dimension	The width of the barcode's smallest element (see Module width).	
Quiet zone	An area free of any printing or marks that precedes the start character of a barcode and follows the stop character. The required minimal size of the quiet zone depends on the barcode type. As a rule, the quiet zone should be ten times the dimension of the module width or at least 1/4 inch (6.5 mm).	
Human Readable Text	This term refers to the entire encoded information of a barcode shown in readable form. It is usually printed below the code. For 2D codes no human readable text is used.	1234567890
Discrete Codes	Each character begins and ends with a bar. The spacing between characters is not part of the code.	
Continuous Code	The spaces between the characters are also part of the code. An example of a continuous code is the Code 2/5 Interleaved.	
Start and Stop Characters	Distinct characters used at the beginning and end of each barcode symbol that provide the scanner with start and stop reading instructions as well as scanning direction.	
Self-checking Code	Self-checking code uses the same pattern for each character. For example, this can be five elements where two of these elements are wide and three are narrow. Any deviation from this pattern would result in an error.	
Check Digit	One or more characters included within the barcode which are used to perform a mathematical check to ensure the accuracy of the scanned data. Check digits are mandatory with certain codes or are even built into the symbology (as for Code-128)	
Bearer Bars	These are bars printed above and below the symbol. The bearer bars are eliminating partial reads (as drawn in the example on the right). Sometimes the complete symbol is surrounded by bearer bars (e.g. ITF-14).	
Substitution Error	Due to reading errors a character is replaced by another during scanning. Substitution errors can be excluded by adding a check digit.	
Synchronizing Bars	These bars are synchronizing the barcode reader. E.g. UPC-A and EAN-13 have synchronizing bars at the beginning, in the middle and at the end of the symbol.	
No-Read	A failure to decode, resulting in no output.	
Misread	The data output of a reader/decoder does not agree with the data encoded in the barcode field. This yields to substitution errors.	

3 ARCHITECTURE OF THE PROJECT

3.1 Architectural Design

Accessing data on a web server's database can be accomplished by using JSON HTTPREQUESTS. But there are limitations imposed by cross-domain scripting. The client applications which requests data from a web server using JSONHTTPREQUESTS can easily retrieve data if the web server is on the same domain.

However, it will have restriction for any requests from the client application to a server in a different domain as Figure 4 shows. Sometimes it is not easy to know what kind of data formats used in external database server. One way is that it can access external database server by using interface based on own web server.

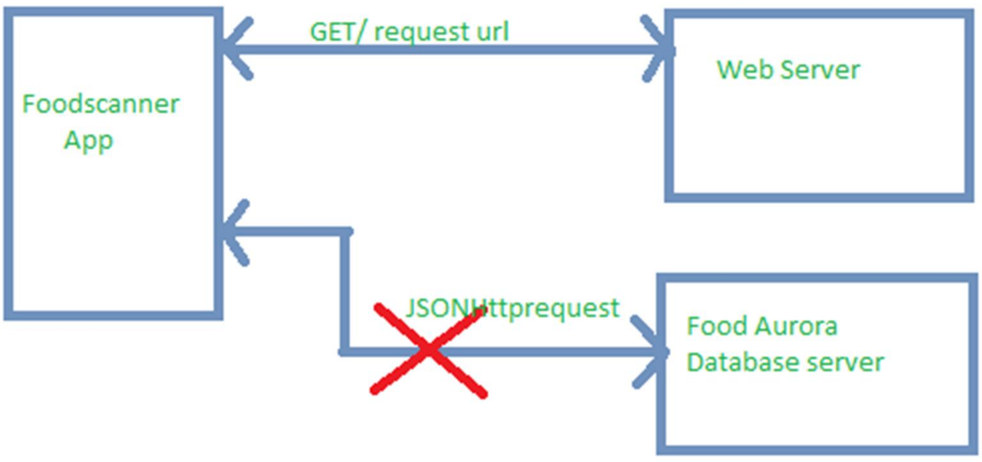


FIGURE 4. Here is a bad way for Accessing database

So the way to design the architecture of the project is formed according to the Model-View-Controller software architecture (MVC).

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts: (MSDN Microsoft ASP.NET MVC Overview 2015.)

- Model –It represent logical structure, the underlying of data in a application. It does not contain the information about the user interface. Model can be used in any data formats, such a model can provide data for multiple views, because the code is applied to the model can be written only once can be reused by the multiple views, so it can reduce repetitive code
- View - which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)

- Controller - The controller is used to respond user input and calls the model and view to complete the user's needs, so when clicking hyperlinks in Web pages and send HTML form, the controller itself does not output anything and do anything. It just receives the request and decides which model to call members to process the request, and then decide which view to display the returned data.

MVC can isolate the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as showing in Figure 5.

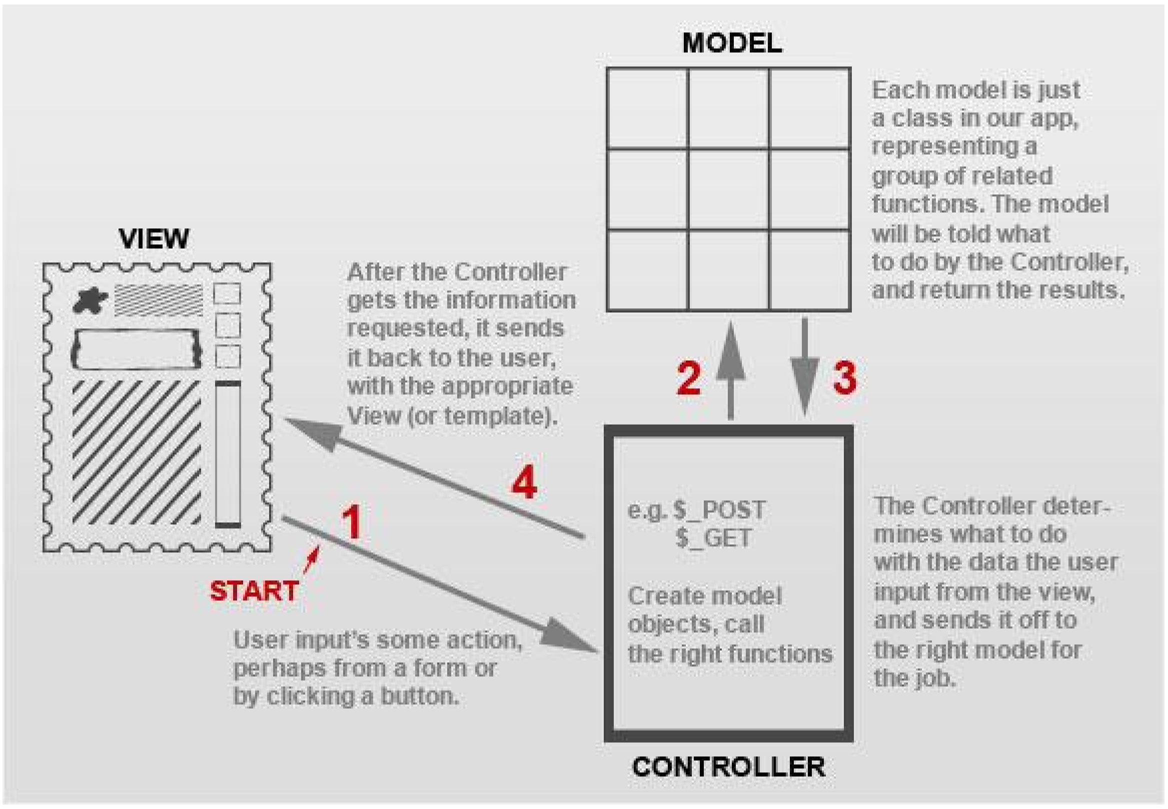

MODEL
Each model is just a class in our app, representing a group of related functions. The model will be told what to do by the Controller, and return the results.
VIEW
After the Controller gets the information requested, it sends it back to the user, with the appropriate View (or template).
2
3
4
1
START
User input's some action, perhaps from a form or by clicking a button.
e.g. $_POST $_GET
Create model objects, call the right functions
The Controller determines what to do with the data the user input from the view, and sends it off to the right model for the job.
CONTROLLER

FIGURE 5. The Instructions of MVC architecture. (RAJESH 2015.)

For this thesis project case, writing some PHP scripts as a web service which reads data from MYSQL and sends as JSON object. The HTTPCLIENT API can invoke this PHP web service and parse the JSON object. The way of it can implement as Figure 6 shows:

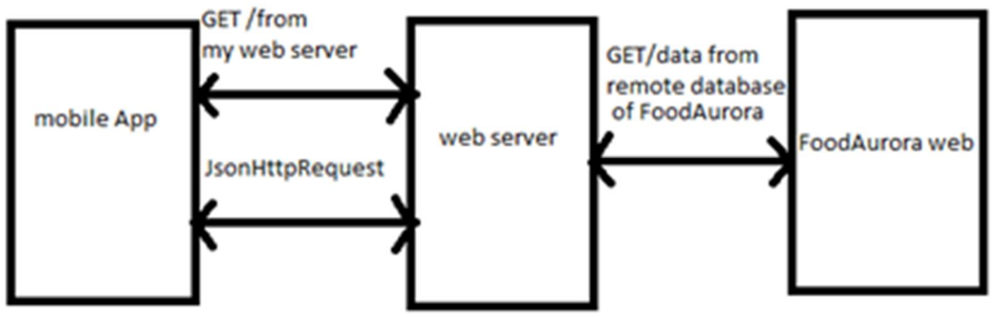


FIGURE 6. The architecture of whole project working

3.2 Web Service

With the introduction of a Web application layer, and from a normal mobile which may has low bandwidth, no enough memory and power client to insert, high power and high bandwidth server with more memory than it needed to run a significant amount of processing - Among them, processing and communications expenses they spend a small part of the environment on the client. Meanwhile, split system, so as to get more control of business rules, the structure of the database, and things outside of version. Once the database so that mobile clients connect directly, the overall design of the delivery system is "married " to a database structure, almost all the changes will break backward compatibility with clients may be reluctant to upgrade their applications. (Tutorials Point, 2015)

Conversely, it is better to add a Web Service between application and external server allows it to interface to a mobile client in a more handled and flexible approach: for example, the old interface in the appropriate position. And it also can add another one to work in Synchronization and then-restructur databse, and it does not need to break a single client. If perfect design principles are made inside when designing the web service based on the mature server-side infrastructure which is put in some place, it can reduce some costs. For example, might get caching and proxy services are free of charge. (Tutorials Point 2015.)

Finally, the new door will be opened to other developers who maybe expose this application to platforms that you could not offer yourself services, and lead good position to your company.

A web service consists of several methods which are advertised for use by the general public. To make it possible for every application to access a web service, some web service protocols are used to support these services, including REST, SOAP, JSON-RPC, JSON-WSP, XML-RPC, and so on. Data can be exchanged in any formats by using a web service, but there are two most popular formats for data exchange as follows:

- XML: Standard format for data exchange among applications, in order to avoid datatype-mismatch problems.
- JavaScript Object Notation (JSON): Text-based open standard for representing data.

In order to make client more lightweight, it is better way to use a web service layer. The advantage is very important to end-users:

- It can reduce payments for users with metered plans by using less bandwidth
- It can increase the battery life by using less CPU on the device
- More efficient and good user experience

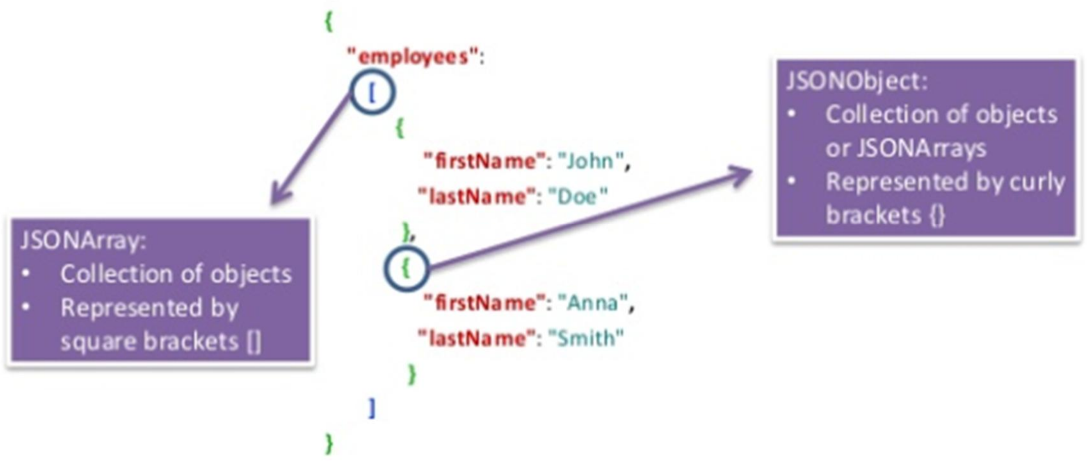
3.3 JSON

To be general, the JSON nodes start witt a curly bracket or a square bracket or. The difference between {and [is, the curly bracket ({) shows JSONObject, however the square bracket ([) represents starting of a JSONArray node. So it should be clear to know how to access these nodes, the suitable method needs to call to access the data.

Firstly, JSON node starts with `[`, then GETJSONARRAY `()` method should be used. if the node starts with `{`, the `GETJSONOBJECT ()` method should be called. (MSDN Microsoft, 2015)

So it is good to know JSONArray and JSONObject, such as one JSONArray may include some JSONObjects. It needs to call different methods above to get data.

In this project, it focuses on the JSON data-exchange format. Data in JSON is represented using simple value pairs, for more-complex data by using the associative arrays. Strings in JSON representation as Figure 7 shows:



```
{
  "employees":
  [
    {
      "firstName": "John",
      "lastName": "Doe"
    },
    {
      "firstName": "Anna",
      "lastName": "Smith"
    }
  ]
}
```

JSONArray:
• Collection of objects
• Represented by square brackets []
JSONObject:
• Collection of objects or JSONArrays
• Represented by curly brackets {}

FIGURE 7. JSONObject and JSONArray. (Wingnity web basics 2014.)

3.4 HTTP

3.4.1 The principles of HTTP

Before accessing the server, it is good to konw how HTTP work. (Apache HttpClient 2008)

- Firstly, build connection between the client and the server.
- Secondly, the client sends a request to the server.
- Thirdly, the server sends a response to the client after receiving a request from the client
- Finally, close the connection after accessing it.

3.4.2 Android HTTPCLIENT

Based on the class of HttpClient, it can support the HTTP as Figure 8 shows. It can be created object of HttpClient to execute and call the method of HTTP GET and HTTP POST.as Fogure 8shows (Zaloni 2011.)

- Initiate the object of HttpClient by the class DefaultHttpClient.
- Creating the object of HttpGet, pass the URL which needs to request to HttpGet by the Construction menthod.
- Sending HttpGet request by calling EXECUTE (), and return the object of HttpReponse.
- It returns the response message by the method of GETENTITY () which implement from HttpResponse. (Developer.android 2015.)

Http clent-server interaction

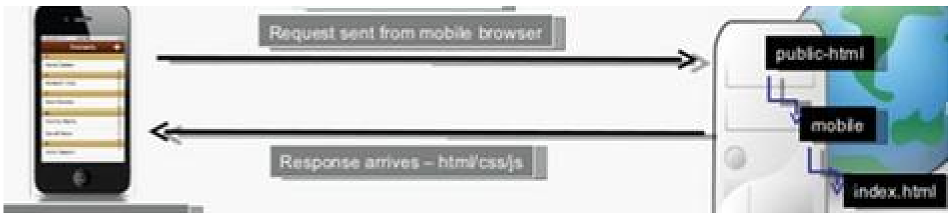

Request sent from mobile browser
public-html
mobile
Response arrives – html/css/js
index.html

FIGURE 8. Client/Server interaction by HTTP.

The Way to implement these to access web showed as fllows. Here two ways which are used to send HTTP REQUEST by calling the GET and POST. The difference between GET and POST: (Developer.android 2015.)

- The parameters which are requested is passed as one part of URL, so that the length of URL is limited less than 2048 char.
- If using HTTP POST to send request, it has no limits of length for url

```
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpEntity httpEntity = null;
HttpResponse httpResponse = null;
```

Checking http request method type

```
 if (method == POST) {
     HttpPost httpPost = new HttpPost(url);
if (params != null) {
         httpPost.setEntity(new UrlEncodedFormEntity(params));
     }
     httpResponse = httpClient.execute(httpPost);

 } else if (method == GET) {

     if (params != null) {
         String paramString = URLEncodedUtils
                 .format(params, "utf-8");
         url += "?" + paramString;
     }
     HttpGet httpGet = new HttpGet (url);
     httpResponse = httpClient.execute(httpGet);
 }
 httpEntity = httpResponse.getEntity();
 response = EntityUtils.toString(httpEntity);

}
```

4 ZXING-OPENSOURCE LIBEARY

ZXing is an open source, 1D / 2D barcode image processing library implemented in Java in a variety of formats, it contains ports which links to other languages. Zxing can be used to complete bar code scanning and decoding based on the phone's built-in camera.

4.1 The formats which ZXING supports

The ZXING library supports the following formats: (ZXING.NET 2014.)

Numeric barcodes:

- EAN-13: the international barcodes which is used for retail products ;
- EAN-8: a barcodes which is specific to small products;
- UPC-A: for Canada and United States;

Bi-dimensional barcodes:

QR code: barcodes which is used for materials management and order confirmation

4.1.1 The ZXING for Android

Firstly, the ZXING library is integrated to the android application, because the whole library is a little complicated, so here some packages given that it needs. The features of each package and files are as follows: (GITHUB 2015.). These Classes are related to bulid barcode scanner on the device. And the

- CaptureActivity-To start an Activity for scanning
- com.google.zxing.client.android.camera----to control the camera on the smartphone
- ViewfinderView----to control the view in camera frame for scanning
- CaptureActivityHandler- to decode the different classes, to call other Thread to decode info it can scan from barcode
- DecodeThread-thread for decoding.

4.2 Some classes for configuring and starting camera

4.2.1 Starting camera on the device

Before starting camera on the device, some permission needs to add to file which is AndroidManifest.xml

```
<uses - permission android: name="android.permission.camera"></uses-permission>
<uses-feature android: name="android.hardware.camera.autofocus" />
<uses-permission android: name="android.permission. vibrate"/>
<uses-feature android: name="android.hardware.camera" />
```

So firstly, it needs to initiate the camera when trigger camera.
The main method which is INITCAMERA is used to initiate the camera

```
@Override
    protected void onResume() {
        super.onResume();
        SurfaceView surfaceView = (SurfaceView) findViewById(R.id.preview_view);
        SurfaceHolder surfaceHolder = surfaceView.getHolder();
        if (hasSurface) {
            initCamera(surfaceHolder);
        } else {
            surfaceHolder.addCallback(this);
            surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        }
        decodeFormats = null;
        characterSet = null;

        playBeep = true;
        AudioManager audioService = (AudioManager) getSystemService(AUDIO_SERVICE);
        if (audioService.getRingerMode() != AudioManager.RINGER_MODE_NORMAL) {
            playBeep = false;
        }
        initBeepSound();
        vibrate = true;
    }
```

4.2.2 The Class of CameraManager

It is a Class which manage camera on the device. It is also the only one class which keeps touch with barcode scanner. The following codes which is used to drive the back camera of the smartphone, and it shows what to do with this method

```
Camera theCamera = camera;
    if (theCamera == null) {
      theCamera = new OpenCameraManager().build().open();
      if (theCamera == null) {
        throw new IOException();
      }
      camera = theCamera;
}
```

4.2.3 Getting the configuration and reading the parameters of camera

```
try {
            configManager.setDesiredCameraParameters(theCamera, false);
    } catch (RuntimeException re) {
```

It will get some exceptions when Driver failed

```
Log.w(TAG, "Camera rejected parameters. Setting only minimal safe-mode parameters");
Log.i(TAG, "Resetting to saved camera params: " + parametersFlattened);

if (parametersFlattened! = null) {
    parameters = theCamera.getParameters();
    parameters.unflatten(parametersFlattened);
    try {
        theCamera.setParameters(parameters);
        configManager.setDesiredCameraParameters(theCamera, true);
    } catch (RuntimeException re2) {
        // Well, darn. Give up
        Log.w(TAG, "Camera rejected even safe-mode parameters! No configuration");
    }
```

4.3 Building the preview and scanning windows

Firstly, the camera has its own interface of the preview, what all need to do is to build a scanning window to lead users put the barcode focus on scanning filed, so that it can read information correctly.

So how to build the scanning window is handled by the class of CaptureActivityHandler which has one method of RESTARTPREVIEWANDDECODE (), then by calling a method of DRAWVIEW-FINDER () in the class of Activity to build the scanning window.

Here are the codes of RESTARTPREVIEWANDDECODE ()

```
private void restartPreviewAndDecode() {
    if ( state==State. SUCCESS) {
      state == State.PREVIEW;
      cameraManager.requestPreviewFrame(decodeThread.getHandler(), R.id.zxinglegacy_decode);
      activity.drawViewfinder();
    }
  }
```

And it will call the method of DRAWVIEWFINDER () in the class of Activity

```
Public void drawViewFinder()
{
viewfindView.drawViewfinder();

}
```

4.4 Screen capture and decode

After the scanning window is build, then scanning the barcode. Firstly, the method of CAMERA.SETONESHOTPREVIEWCALLBACK () is to check and trigger the event for screen capture. Then, the scanning event is monitored by the CAMERA.PREVIEWCALLBACK ().

```
final class PreviewCallback implements Camera. PreviewCallback {

  private static final String TAG = PreviewCallback.class.getSimpleName();

  private final CameraConfigurationManager configManager;
  private Handler previewHandler;
  private int previewMessage;

  PreviewCallback(CameraConfigurationManager configManager) {
    this.configManager = configManager;
  }
  void setHandler(Handler previewHandler, int previewMessage) {
    this.previewHandler = previewHandler;
    this.previewMessage = previewMessage;
  }
```

The following Method of ONPREVIEWFRAME () is used to send decode message to the class of DecodeHandler when the preview interface is opened, at the same time, the byte array of DATA [] is passed to the callback function. The code is implemented as follows:

```
  @Override
  public void onPreviewFrame (byte [] data, Camera camera) {
    Point cameraResolution = configManager.getCameraResolution ();
    Handler thePreviewHandler = previewHandler;
    if (cameraResolution! = null && thePreviewHandler! = null) {
      Message message = thePreviewHandler.obtainMessage (previewMessage, cameraResolution.x,
          cameraResolution.y, data);
      message.sendToTarget ();
      previewHandler = null;
    } else {
      Log.d (TAG, "Got preview callback, but no handler or resolution available");
    }
  }
```

Finally, after get message for decoding, the class of DecodeHandler call its function of DECODE ().

```
private void decode (byte [] data, int width, int height) {
    long start = System.currentTimeMillis ();
    Result rawResult = null;
    PlanarYUVLuminanceSource source = activity.getCameraManager ().buildLuminanceSource (data,
width, height);
    if (source! = null) {
      BinaryBitmap bitmap = new BinaryBitmap (new HybridBinarizer (source));
      try {
        rawResult = multiFormatReader.decodeWithState (bitmap);
      } catch (Exception e) {

      } finally {
        multiFormatReader.reset ();
      }
    }

    Handler handler = activity.getHandler ();
    if (rawResul! = null) {

      long end = System.currentTimeMillis ();
      Log.d (TAG, "Found barcode in " + (end - start) + " ms");
      if (handler! = null) {
        Message message = Message.obtain (handler, R.id.zxinglegacy_decode_succeeded, rawRe-
sult);
        Bundle bundle = new Bundle ();
        bundleThumbnail (source, bundle);
        message.setData (bundle);
        message.sendToTarget ();
      }
    } else {
      if (handler! = null) {
        Message message = Message.obtain (handler, R.id.zxinglegacy_decode_failed);
        message. sendToTarget ();
      }
    }
  }
```

4.5 Decoding results handled

After, the decoding process is finished, then the method of DECODE () OF DecodeHandler will send messages to the class of CaptureActivityHandler, If decoding successfully, then the method of HANDLEDECODE () in the class of CaptureActivity will process the scanning results according to the Classification.

At first, Parsing rawResult, then creating corresponding ResultHandler according to different type of result.

```
  public void handleDecode (Result rawResult, Bitmap barcode, float scaleFac     tor) {
    inactivityTimer.onActivity ();
    lastResult = rawResult;

    boolean fromLiveScan = barcode! = null;
   if (fromLiveScan) {
```

Then not from history, so beep/vibrate and we have an image to draw on

```
   beepManager.playBeepSoundAndVibrate ();
   drawResultPoints (barcode, scaleFactor, rawResult);
   }

  handleDecodeExternally (rawResult, barcode);
    }
```

Then Call these two methods which are HandleDecodeInternally and handleDecodeExternally to process the ResultHandler

4.6 Display the results

CameraManager must be initialized here, not in onCreate(). This is necessary because we don't want to open the camera driver and measure the screen size if we're going to show the help on first launch. That lead to bugs where the scanning rectangle was the wrong size and partially

```
  cameraManager = new CameraManager(getApplication());

  viewfinderView = (ViewfinderView) findViewById(R.id.zxinglegacy_viewfinder_view);
  viewfinderView.setCameraManager(cameraManager);

  resultView = findViewById(R.id.zxinglegacy_result_view);
  statusView = (TextView) findViewById(R.id.zxinglegacy_status_view);

  handler = null;
  lastResult = null;

  resetStatusView();

  SurfaceView surfaceView = (SurfaceView) findViewById(R.id.zxinglegacy_preview_view);
  SurfaceHolder surfaceHolder = surfaceView.getHolder();
if (hasSurface) {
```

The activity was paused , it does not mean it will be stopped, so the surface still be keep there. Thus, surfaceCreated () does not be called, so initial the camera is here.

```
      initCamera(surfaceHolder);
} else {

      Install the callback and wait for surfaceCreated() to init the camera.

      surfaceHolder.addCallback(this);
      surfaceHolder.setType(SurfaceHolder. SURFACE_TYPE_PUSH_BUFFERS);
    }

    beepManager.updatePrefs();
    ambientLightManager.start(cameraManager);

    inactivityTimer.onResume();

    Intent intent = getIntent();

    SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);

    source = IntentSource.NONE;
    decodeFormats = null;
    characterSet = null;

    if (intent! = null) {

      String action = intent.getAction();
      String dataString = intent.getDataString();

      if (Intents.Scan.ACTION.equals(action)) {
```

Scan the formats the INTENT requested, and after reading barcodes ,then return the result to the calling activity.

```
        source = IntentSource. NATIVE_APP_INTENT;
        decodeFormats = DecodeFormatManager.parseDecodeFormats(intent);
        decodeHints = DecodeHintManager.parseDecodeHints(intent);

   if (intent.hasExtra(Intents.Scan.WIDTH)&&intent.hasExtra(Intents.Scan.HEIGHT)) {
         int width = intent.getIntExtra(Intents. Scan. WIDTH, 0);
         int height = intent.getIntExtra(Intents. Scan. HEIGHT, 0);
         if (width > 0 & & height > 0) {
           cameraManager.setManualFramingRect(width, height);
        }
      }
      String customPromptMessage=intent.getStringExtra(Intents.Scan.PROMPT_MESSAGE);
      if ( customPromptMessage! = null) {
        statusView.setText (customPromptMessage);
      }

    } else if (dataString! = null &&
            dataString.contains (PRODUCT_SEARCH_URL_PREFIX) &&
            dataString.contains(PRODUCT_SEARCH_URL_SUFFIX)) {
```

Scan products and send the result to mobile Product Search.

```
      source = IntentSource. PRODUCT_SEARCH_LINK;
      sourceUrl = dataString;
      decodeFormats = DecodeFormatManager.PRODUCT_FORMATS;

    } else if (isZXingURL(dataString)) {

      source = IntentSource. ZXING _LINK;
      sourceUrl = dataString;
      Uri inputUri = Uri.parse(dataString);
      decodeFormats = DecodeFormatManager.parseDecodeFormats(inputUri);
      decodeHints = DecodeHintManager.parseDecodeHints(inputUri);

  }

    characterSet = intent.getStringExtra(Intents.Scan.CHARACTER_SET);

   }
```

Here it scan the formats which is the intent requested, and return the result to the calling activity.

```
                source = IntentSource.NATIVE_APP_INTENT;
                decodeFormats = DecodeFormatManager.parseDecodeFormats(intent);
                decodeHints = DecodeHintManager.parseDecodeHints(intent);
                if (intent.hasExtra(Intents. Scan.WIDTH) && in-
tent.hasExtra(Intents.Scan.HEIGHT)) {
          int width = intent.getIntExtra(Intents. Scan. WIDTH, 0);
          int height = intent.getIntExtra(Intents.¨Scan. HEIGHT, 0);
          if (width > 0 && height > 0) {
            cameraManager.setManualFramingRect(width, height);
          }
        }
        if (intent.hasExtra ( Intents. Scan.CAMERA_ID)) {
          int cameraId = intent.getIntExtra ( Intents. Scan. CAMERA_ID, -1);
          if (cameraId >= 0) {
            cameraManager.setManualCameraId(cameraId);
          }
        }

        String customPromptMessage = intent.getStringExtra(Intents. ScanP-
ROMPT_MESSAGE);
        if (customPromptMessage! = null) {
          statusView.setText(customPromptMessage);
        }
      } else if (dataString != null &&
                 dataString.contains("http://www.google") &&
                 dataString.contains("/m/products/scan")) {
 Scan only products and send the result to mobile Product Search.
        source = IntentSource.PRODUCT_SEARCH_LINK;
        sourceUrl = dataString;
        decodeFormats = DecodeFormatManager. PRODUCT_FORMATS.

      }
```

Barcode will save the analyzed results to Intent When “source == IntentSource.NATIVE_APP_INTENT”. Then return to the object which belongs to Application display interface.

5 IMPLEMENTATION OF THE PROJECT

5.1 Android Application

The main developing tools that were used are:

- Android Studio. (Android Developer, 2105)
- Dreamweaver for writing PHP scripts acting as a proxy.
- XAMMP for local MySQL server to host databases and data. (Xampp, 2015)

The main components of the applications are:

- The ZXING library (imported in the local resources of the application).
- The core of ZXING. (GITHUB, 2015)
- Api connector which is used to build connection between the application and the database server by using HttpClient.

5.2 Android application development

Firstly, the Android project is created and named “Foodscanner”, and given the main layout as Figure 9 shows, when pressing the scan tag button, this button will trigger and start CaptureActivity to get the camera interface, and then it will read the barcode to get information on the barcode, such as the contents on Bar code, the type on Bar code.

5.2.1 Activity_main.layout

Now, it needs to create a layout to show the interface of the main activity as Figure 9 shows.

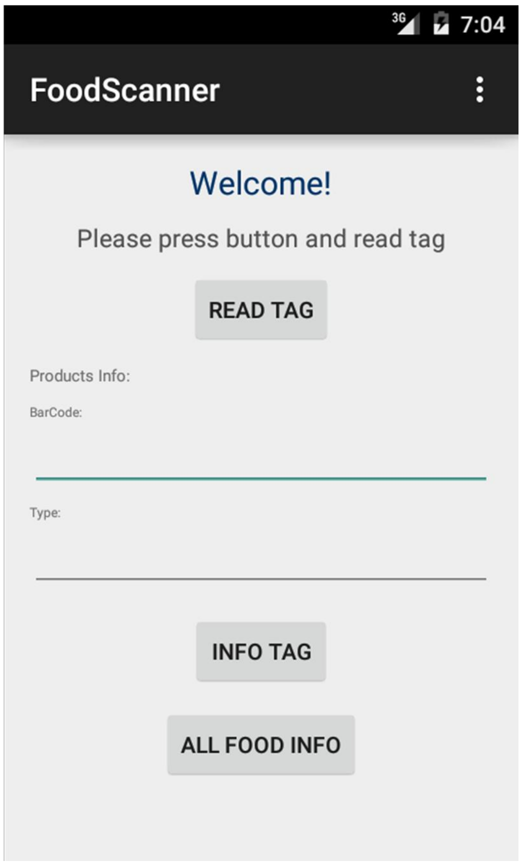


FIGURE 9. The main layout when launching

5.2.2 The implementation of codes

Here is the button event to trigger an activity of SimpleScannerActivity to start the camera on the device.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        scanbtn = (Button) findViewById(R.id.button1);

         scanbtn.setOnClickListener (newView.OnClickListener () {

            @Override
            public void onClick(View v) {

                Intent intent = new Intent(MainActivity.this, SimpleScannerActivity.class);
                startActivity(intent);
            }
        });
}
```

When SimpleScannerActivity is triggered, it will start the camera on the device, and then basic scanning camera frame is made as Figure 10 shows. After it scan the barcode, it will get the contents on barcode and the type of barcode shuch as EAN_13. It also shows them in the following area as Figure 9 shows which are “Barcode” and “Type”.

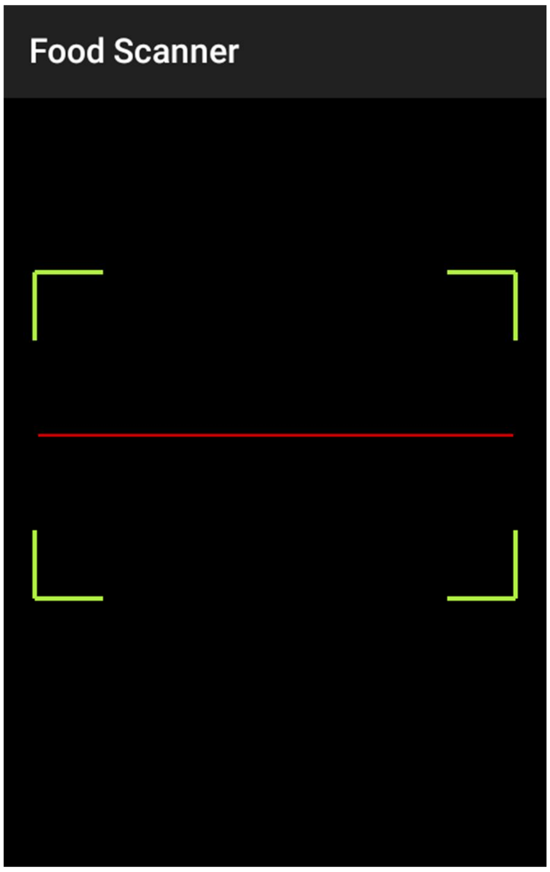


FIGURE 10. Camera frame for scanning barcode

5.2.3 Testing project

After finishing the whole project, the project was debugged. Here are some results as follows. One barcode attached on a product as Figure 11 shows, it is used for testing.

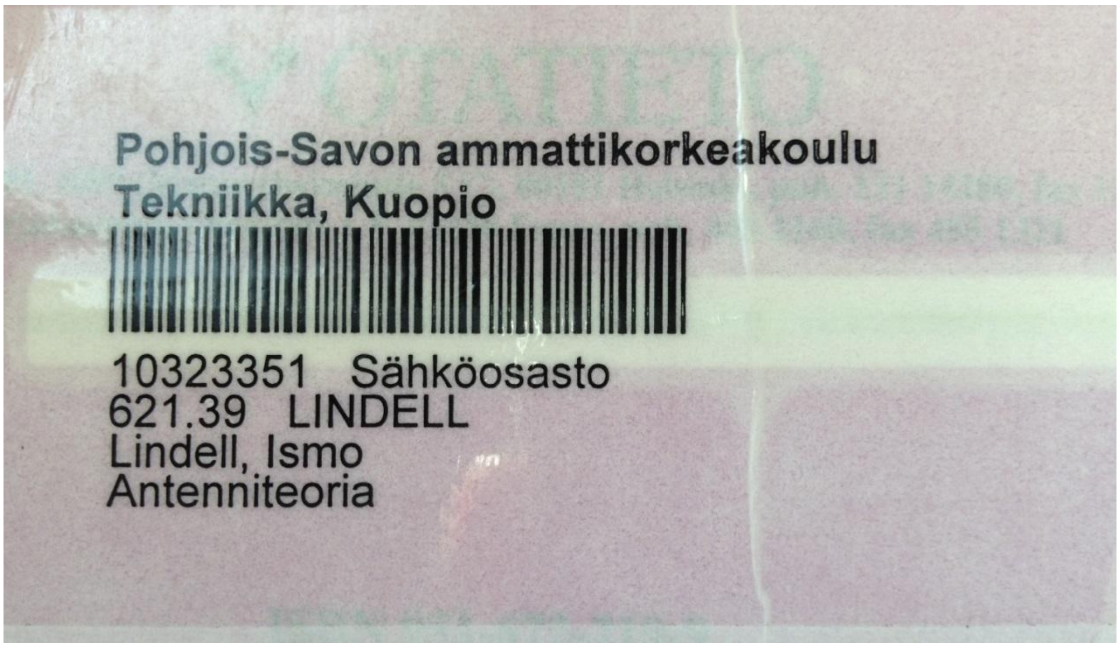

Pohjois-Savon ammattikorkeakoulu
Tekniikka, Kuopio
10323351 Sähköosasto
621.39 LINDELL
Lindell, Ismo
Antenniteoria

FIGURE 11. Barcode for testing

Testing application on a real device, and pressing the following button (READTAG) as Figure 12 shows.

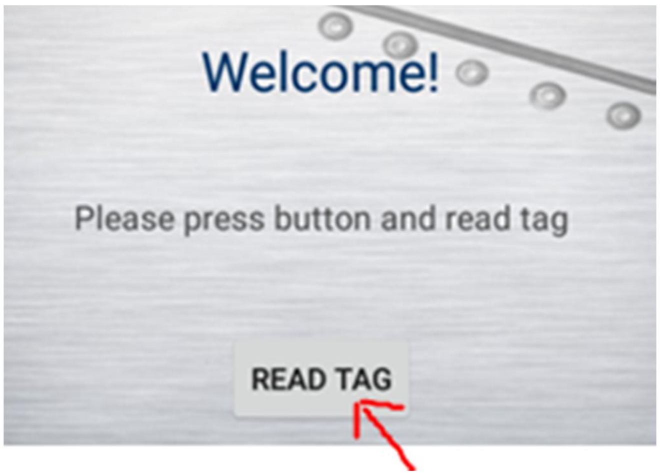


FIGURE 12. Press button to start scan

Here getting the results the following two pictures as Figure 13 and Figure 14 shows:

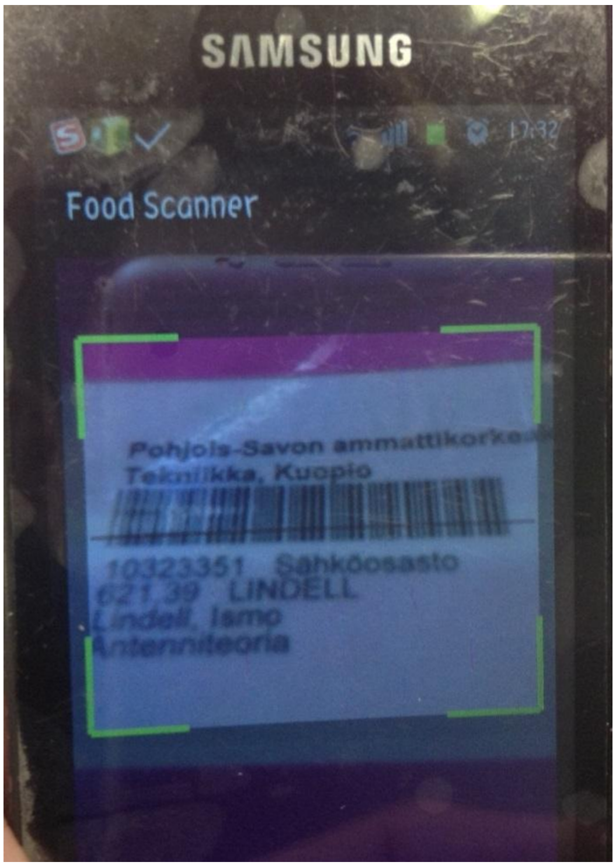
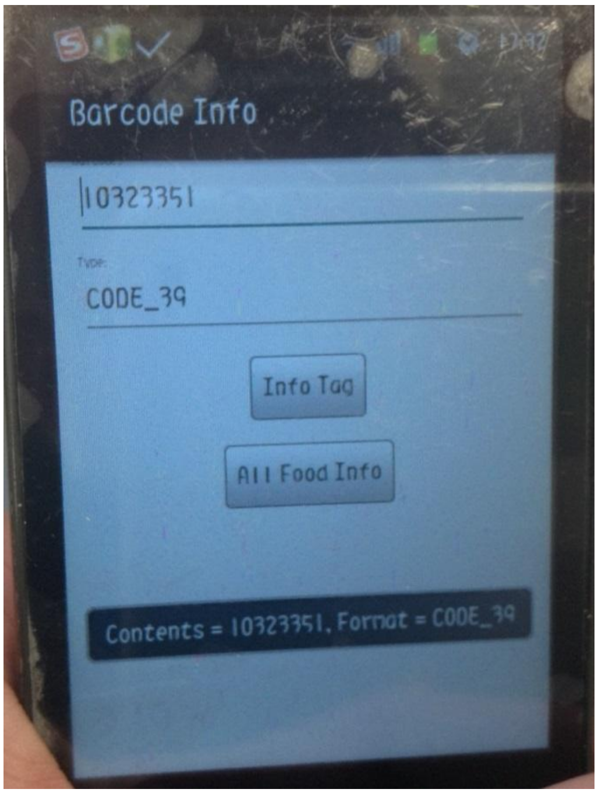


FIGURE 13. Scanning barcode



10323351

Type:

CODE_39

Contents = 10323351, Format = CODE_39

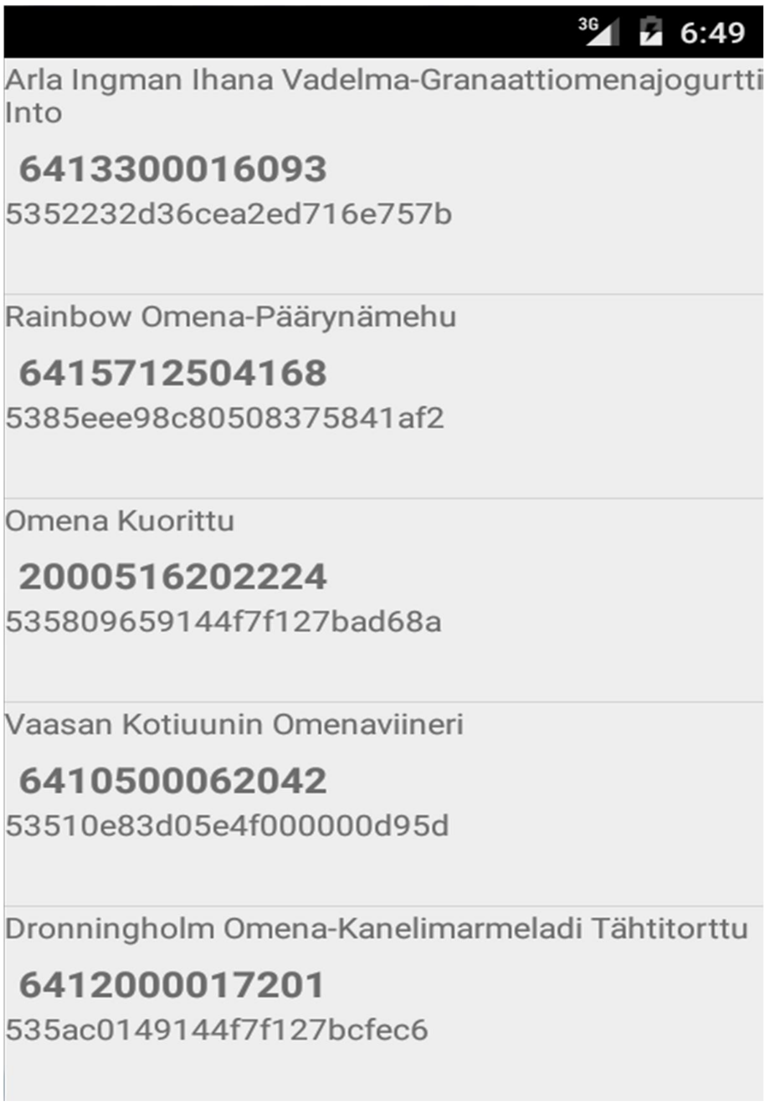
FIGURE 14. Scanning results from testing barcode

5.2.4 Accessing the Food Aurora Web Server

Before connecting remote database server, the permission accessing Internet should be added to file (AndroidManifest.xml)

<uses-permission android:name="android.permission.INTERNET"/>

Here some parts of web service to get all food information from Food aurora web by API they offer It can show the all food in the ListView as Figure 15 shows.



Arla Ingman Ihana Vadelma-Granaattiomenajogurtti Into
6413300016093
5352232d36cea2ed716e757b

Rainbow Omena-Päärynämehu
6415712504168
5385eee98c80508375841af2

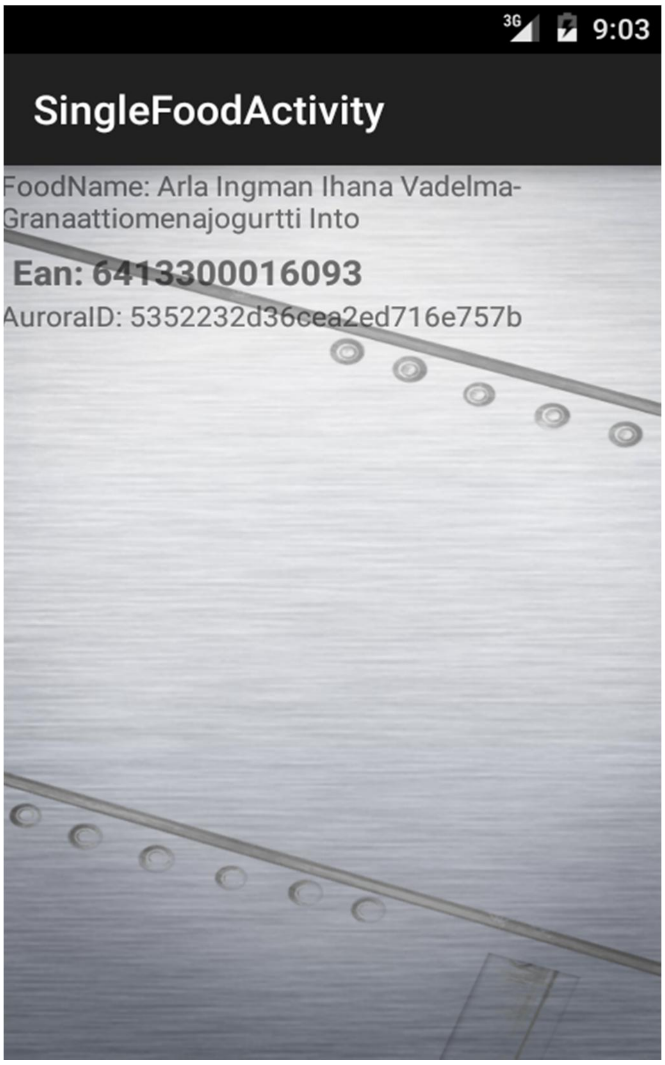
Omena Kuorittu
2000516202224
535809659144f7f127bad68a

Vaasan Kotiuunin Omenaviineri
6410500062042
53510e83d05e4f000000d95d

Dronningholm Omena-Kanelimarmeladi Tähtitorttu
6412000017201
535ac0149144f7f127bcfec6

FIGURE 15. The listview for all food information

When the each item is clicked, it can show more information about each food as Figure 16 shows.



FoodName: Arla Ingman Ihana Vadelma-Granaattiomenajogurtti Into

Ean: 6413300016093

AuroraID: 5352232d36cea2ed716e757b

FIGURE 16. The Single food details

6 IMPROVING THE PROJECT

Based on this basic project, there should be a lot of things that can be implemented. Such as adding the user registration and login, and sharing the information on food to some social networks. It would be better to improve the resolution of the camera, autofocus, and or macro lens to obtain accurate and fast decoding. In that case it can increase user experience if visual clues on how to align the barcode image for the best capture and barcode reading are given. Besides adding new hardware, the image capture of the camera could also be improved. Except for the camera, some parts also need to challenge such as power consumption for the device, and also the efficiency of algorithms needs to improve to make sure that the application can run stably. If a much bigger food database server can be offered, some functions can be added, for example comparing some information and prices of similar food products online when scanning some barcodes attached on food products.

7 CONCLUSION

The application can scan barcodes successfully and access the web server on FoodAurora. It was a very good experience to get more practical skills through this project. More professional skills are gotten to know, such as the web service, how to use JSON to access the external database with the help of PHP scripts, and also get to know how to integrate the library project to the application.

With the basic design of project architecture, the food scanner works very close to how it is expected. Because the provided camera has low resolution, even though it can autofocus, however it was often difficult to capture barcode image quickly and determine the exact cause of problems. But anyway, after extensive testing it works with good results. Thanks to efficient images conversion and decoding processes, the whole process runs predictably within reasonable time constraints; almost in a few seconds. Although decoding often takes up the majority of the process, the time it takes is usually constant. Thus, the network access may be considered. If the network can not be connected stably, it may produce uncontrollable results. Barcode decoding on the device should be definitely possible with a perfect implementation. Finally, the quality of the camera is also to be considered.

REFERENCES

Android Developer, Android Studio Overview 2014 [Retrieved 2015-03-02]
Available at:
http://developer.android.com/tools/studio/index.html

Apache HTTPClient Tutorials 2008 [Retrieved 2015-04-22]
Available at:
http://hc.apache.org/httpclient-3.x/tutorial.html

Developer.android, AndroidHttpClient 2014 [Retrieved 2015-05-01]
Available at:
http://developer.android.com/reference/android/net/http/AndroidHttpClient.html

GITHUB, ZXING CORE 2013 [Retrieved 2015-04-22]
Available at:
https://github.com/zxing/zxing/releases

GITHUB, ZXING Resourse 2014 [Retrieved 12015-04-17]
Available at:
https://github.com/zxing/zxing

Joseph Khan 2011 [Retrieved 2015-04-30]
Available at:
https://www.google.fi/search?hl=fi&tbm=isch&q=mobile+app+httpclient+server+zaloni&ei=WKNZ
VaG-
WJcOusQHd04HACg#imgrc=Z3LPhn_E1w8BPM%253A%3BZBqaAW1QnmcMLM%3Bhttp%253A%25
2F%252Fimage.slidesharecdn.com%252Fsenchatouch-110817070112-
phpapp02%252F95%252Fintroduction-to-sencha-touch-developing-web-applications-for-mobile-
devices-9-
728.jpg%253Fcb%253D1325802509%3Bhttp%253A%252F%252Fwww.slideshare.net%252Fjsphkh
an%252Fintroduction-to-sencha-touch-developing-web-applications-for-mobile-
devices%3B728%3B546

MSDN Microsoft ASP.NET MVC Overview 2015 [Retrieved 2015-04-10]
Available at:
https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx

MSDN Microsoft, system JSON 2014 [Retrieved 2015-04-12]
Available at:
https://msdn.microsoft.com/en-us/library/system.json%28v=vs.95%29.aspx

Omniplanar, Barcode Definition 2013 [Retrieved 2015-04-30]
Available at:
http://www.omniplanar.com/barcode-definitions.php

RAJESH Paul, (MVC) Coding Structure 2013 [Retrieved 2015-04-05]
Available at:
https://rajesh38.wordpress.com/2013/09/25/a-detailed-overview-of-the-model-view-controller-mvc-coding-structure/

TEC-IT, Barcode Software Barcode Overview 2015 [Retrieved 2015-04-05]
Available at:
http://www.tec-it.com/Download/PDF/Barcode_Reference_EN.pdf

Tutorials Point, Web Services Resources [Retrieved 2015-04-20]
Available at:
http://www.tutorialspoint.com/webservices/web_services_quick_guide.htm

Wikipedia CCD 2015 [Retrieved 2015-04-30]
Available at:
http://en.wikipedia.org/wiki/Charge-coupled_device

Wikipedia, GITN 2013 [Retrieved 2015-04-29]
Available at:
http://en.wikipedia.org/wiki/Internet_of_Things

Wikipedia, IOT 2014 [Retrieved 2015-04-30]
Available at:
http://en.wikipedia.org/wiki/Internet_of_Things

Wingnity web basics 2014 [Retrieved 2015-04-29]
Available at:
http://www.slideshare.net/wingnity/android-week-5

ZXING.NET 2014 [Retrieved 2015-04-26]
Available at:
https://zxingnet.codeplex.com/

XAMPP, XAMPP Tutorial 2014 [Retrieved 2015-04-27]
Available at:
https://blog.udemy.com/xampp-tutorial/