MAJOR PROJECT-II DISSERTATION

Android Application for Restaurant Ordering and Home Delivery

by

SupreetSingh(R780209045)
Hemant Mendi Ratta(R780209041)
Akshay Thapliyal(R780209005)
Naman Garg(R780209019)
B.Tech CSE-IVyr-VIIISem
(2009-2013 batch)

Supervised by:

Dr. Ajay Shanker Singh
Assistant Professor, Dept. Of CSE
(Senior Scale)

Submitted to the Department
of
Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Technology

University of Petroleum & Energy Studies, Dehradun

P.O. Bidholi Via-Prem Nagar, Dehradun, INDIA-248007

CERTIFICATE

This is to certify that the Project entitled

RESTAURANT APPLICATION FOR ANDROID MOBILE PHONES

Submitted by:

Name	Roll No.
Hemant Mendiratta	R780209041
AkshayThapliyal	R780209005
Naman Garg	R780209019
Supreet Singh	R780209045

For the progress report of the requirements of the course **Major Project** of **Bachelor of Technology in Department of Computer Science and Engineering** degree of **University of Petroleum & Energy Studies**, Dehradun embodies the confide work done by above students under my/our supervision.The minute things may be changed in final version.

Signature of Supervisor
Mr. Ajay Shanker Singh
(Assistant Professor, Dept. of C.S.E.(S.G.).)

Date: 24/4/2013

ACKNOWLEDGEMENT

Words are often too weak to express our feelings of indebtedness to ones benefactors. The same difficulty haunts to me in penning down the deep sense of gratitude.

We take this opportunity to thank the people at UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, who so generously contributed to our requirements and gathering work with their unfailing support. We appreciate very much having help from Mr. Vishal Kaushik, who provided all required resources and made arrangements that every trainee needs from his/her guide . We are grateful to Prof. Dr. MANISH PRATEEK, Head of Department (CSE) and Prof. Ajay Shankar(Course Coordinator), who provided many helpful suggestions and who had made this project possible by his support on technical points during the critical phases of this project. We humbly acknowledge our profound debt of gratitude to Mr. Ajay Shankar Singh, project coordinator, whose clear vision in the field of IT helped us in forming clear concepts and facilitating our work from time to time.

Our deepest appreciation to the faculty members of "Department of Computer Science and Engineering" who gave us such a precious chance to make this project and who inspired us from time to time for doing such type of activities. Last but not the least I thank to my friends and family for their support and encouragement, which enables us to work efficiently and confidently.

SUBMITTED BY-

HemantMendiratta	R780209041
AkshayThapliyal	R780209005
NamanGarg	R780209019
Supreet Singh	R780209045

ABSTRACT

Android application for restaurants acts as medium through which people can order food items of any choice available in restaurants and restaurant can receive the orders and deliver them conveniently. More importantly with this application users can order food from anywhere and any place but of course the restaurant should allow the delivery at that place.

Here we intend to build an automated system for facilitating online ordering and home delivery. To achieve this our system is developed on android which is popular operating system for mobile phones and will be dominating for coming years. The system consists of two applications one for user side or for use by masses to order food items and the second for restaurant side where orders placed are received and delivered to the users or people.

In user side application, list of food items with associated categories are there from which the items to be had by the user can be selected .The application will contain the option to place the order and send the order along with address and contact details of the place where the order has to be delivered.

In the restaurant side application, the order is received.This application will have the capability of auto respond and thereafter depending on restaurant itmes availability notification is sent to the user i.e.user side application containing unavailability of items or deliver details ,bill cost and duration of home delivery.

This application is an effort by us to bring android technology more closer to people's life.

TABLE OF CONTENTS

CHAPTER 1

1.0 Introduction

1.1 An Overview of the Existing System

Visiting restaurants or ordering food is a part of every individual life.Good standard has to be maintained by the restaurants to attract more customers.In today's world, mobile applications have brought a revolution and allow to achieve variety of tasks like ordering foods, booking tickets,buying apparels,etc in no time.Changing Technology demands its use in this area too to provide masses with more convenient means.To develop this system android technology is used.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools necessary to begin developing applications on the Android platform using the Java programming language.**Android** is a Linux-based open source operating system for mobile devices such as Smartphone's and tablet computers.

Google led Open Handset Alliance has initiated the development of Droid in 2008.

The short story of Android is listed below:

- Programmed in: C (core),Java (UI)
- Source model : Open Source
- Initial Release: September 20, 2008
- Latest Release: 4.2 (Jelly Bean)/ October 29 , 2012
- Official website: www.android.com

1.2 An Overview of the Proposed System

The system comprises of two android applications one for the user side(customer) to order food items and other for the restaurant side for accepting orders placed by customers.

User side places the order and restaurant acknowledges it and sends the notification to the user side for delivering their order.

1.3 Need for proposed system

- An android application will help people in a long way to get eateries from anyplace (convenient)and at any time they need.
- Restaurants too will be using android devices having the restaurant application with which customers will be ordering.
- This scenario gives rise to our problem:
- Developing an Android Application for Restaurants and Hotels to facilitate home delivery and instant ordering.

1.4 Introduction to Area of selection

For developing restaurant application, an Eclipse (IDE), Android SDK,jdk is used.

- First of all Install jre then install eclipse i.e.(eclipse-SDK-3.6.2-win32) after that install jdk (cnet2_jdk-6-windows-i586_zip) .
- Install android tools (SDK manager and AVD manager) in ECLIPSE.
- Now,Eclipse provided the feature to develop android projects.
- The procedure to use the emulator camera is as follows:
  - Firstly,sd card needs to added to the emulator.
  - SD card is added to the directory.
  - This sd card is addcd to the AVD whilc launching it.

CHAPTER 2

2.0 Domain Analysis

2.1 Problem Analysis

2.1.1 Information Gathering

In today's advancing world everyone wants that their day to day activities happen in a convenient way.These activities include buying clothes,booking tickets and ordering food items.

Our System will provide one touch access to the user for ordering food anytime and any place.

2.1.2 Problem Definition

- An android application will help people in a long way to get eateries from any place(convenient)and at any time they need.
- Restaurants too will be using android devices having the restaurant application with which customers will be ordering.
- This scenario gives rise to our problem:
- Developing an Android Application for Restaurants and Hotels to facilitate home delivery and instant ordering .

2.2 Features of Proposed System

2.2.1 Purpose of the System

The purpose of the system is to develop two applications, one for the client side and another for the restaurant side. The client side application will have course menu of the restaurant, through which the user can select the desired food items and can order it

through the application. And the restaurant side will receive the orders from the user application and respond to it that "the order will be delivered soon".

This system helps people to order food at any time from anyplace. This system is based on "Cash On Delivery", means the payment will be done by the user once the order is received. The "payment gateway" technology is not deployed in the application.

Basic Feature:

- Application can be used to place orders by the users to the restaurants.
- Restaurant application will receive the order placed and send notification to the user.

Enhanced Feature:

- The application will have online payment mode instead of Cash On Delivery.

2.2.2Design Methodology

Activity Diagram

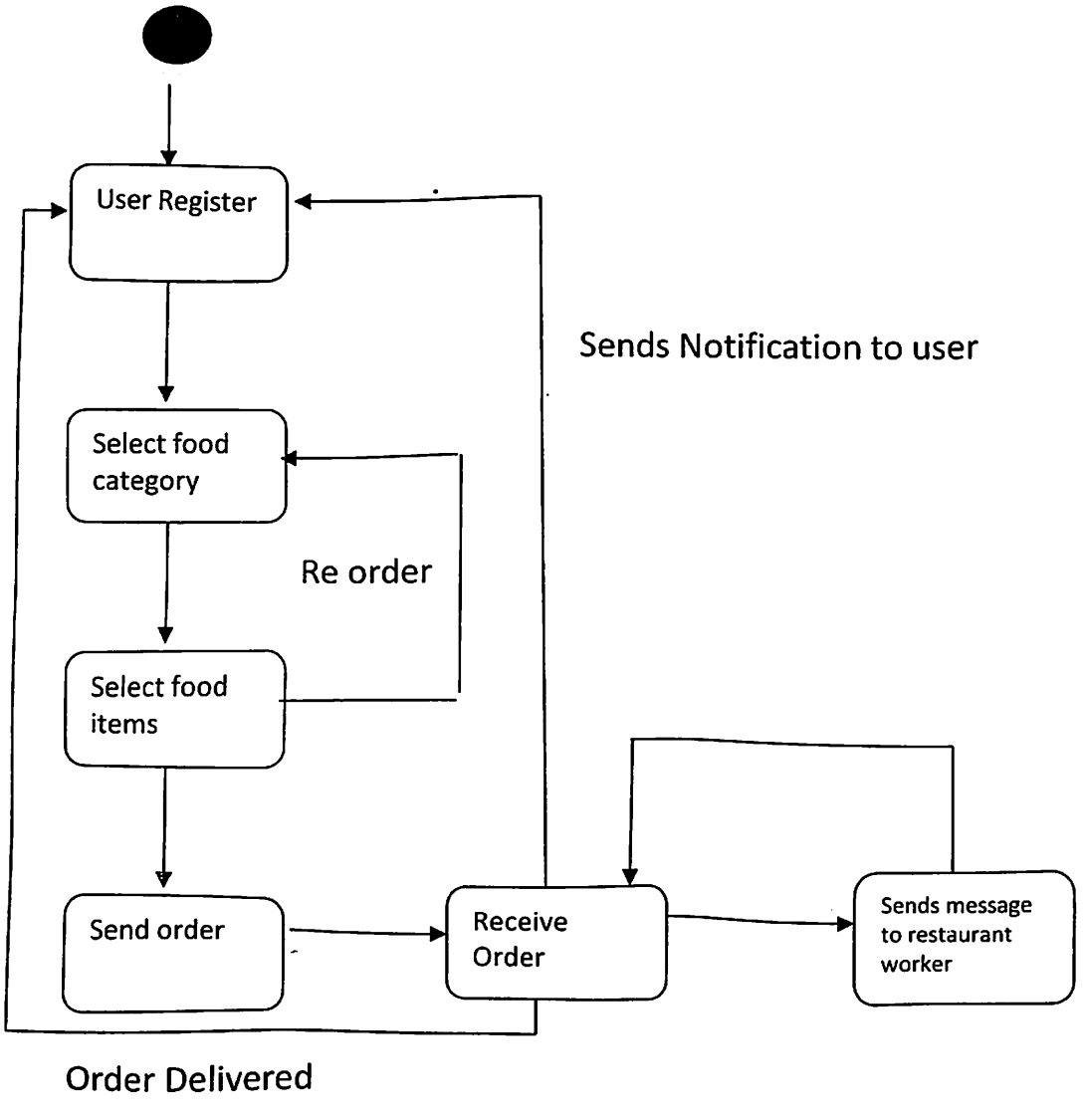

Figure 1:Activity Diagram

Class Diagram

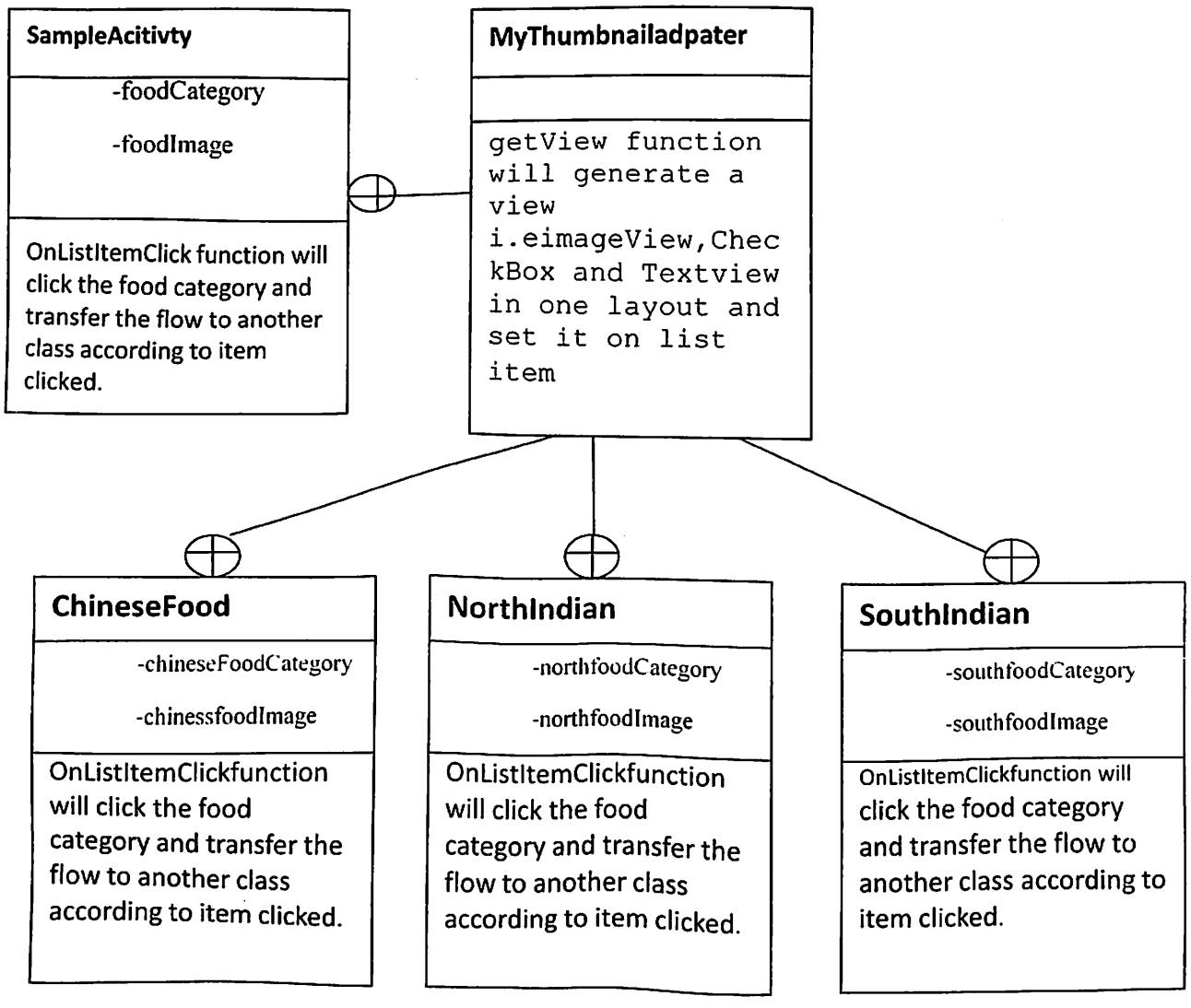

Figure 2:Class Diagram

2.2.3 Scope of the System

The system works for the smart phones with android operating system. The application requires the message pack otherwise the charges will be deducted from main balance.

The application will not work for Symbian,Blackberry,IOS or BADA operating systems.

CHAPTER 3

3.0Requirement Specification

Technical Requirements

One: JDK (latest Java Development Kit 1.6/1.7).

Two: Install Android Development Tools Plug-in(SDK manager and AVD manager) inECLIPSE.

Three: Now, EclipseIDE provides the platform with all necessary and required features and functionality to develop android projects.

Four: SqliteManagerplug-in is needed to enable the local database.

3.1.1. Input Requirements

Input is the valid order which consists of at least on item which is sent by the user to restaurant and the valid address where the order is to be sent.

3.1.2. Output Requirements

The order received by the restaurant is preparedby the restaurant and delivered to the user at his/her address.

3.1.3. Computational Requirements

- Synchronization between the two applications user and the restaurant.Which is done by the help of a server.

3.1.4. Storage Requirements

To store the orders placed by user we need to create a storage system within the restaurant side application means the SD card needs to be created. Following are the steps to create a virtual SD card inside the Emulator:-

To achieve variety of tasks sd card needs to be made for android emulator so that applications can be tested on it.The method to make an SD card for android emulator is:

- Navigate to the ***tools*** folder in ***android-sdks*** directory present in the path(C:\Users\user_account\android-sdks\tools) in the command prompt using cd
- Here give the following command

mksdcard 128M my128MbCard

mksdcard is the command to execute the mksdcard exe file which creates the sd-card.128Mb is the memory to be given to sd-card and my128MbCard is the name given to the sd-card.

- A file with the specified name and size is created in the ***tools*** folder which needs to be added to the android emulator.
- This file is added to the emulator by going to the Android Virtual Device(AVD) Manager and choosing the edit option for already created avd device and browsing the created file by the above command in the field of sd card and click on Edit AVD.

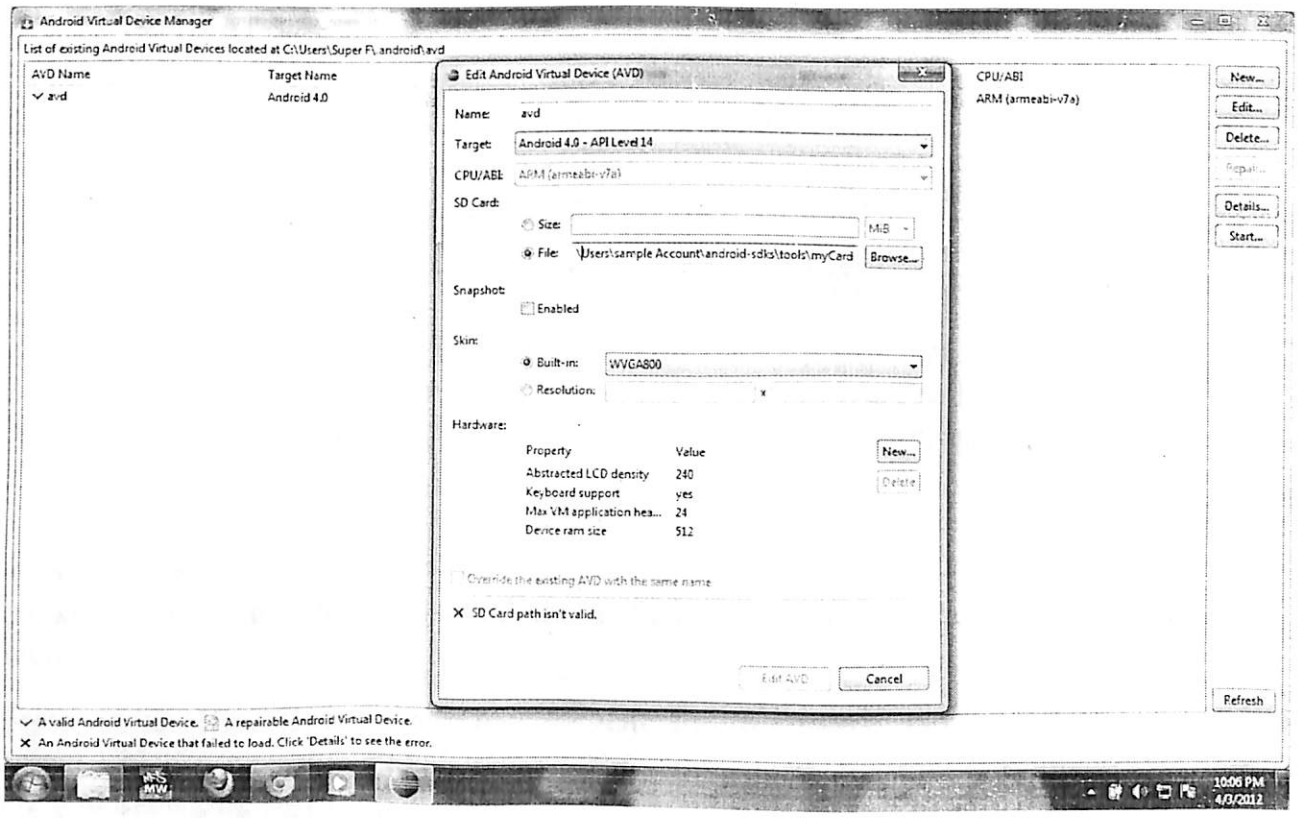

Figure 3:Adding sd card file to emulator

In this sdcard a local database is created with Sqlite to store the orders.

3.2. Hardware Requirements

A device capable of running ECLIPSE IDE and JDK. The performance criteria for such system is given below :

Processor: Dual Core

RAM: 2 GB

Android Smartphone is required to test the Application live.

3.3. Software Requirements

Only three things are needed to setup for building android applications. These are JDK(Java Development Kit) , Eclipse IDE and ADT(Android Development Tool) Plug-in.

1. JDK (Java Development Kit)

Download and setup any JDK (latest Java Development Kit 1.6/1.7).The JDK has as its primary components a collection of programming tools for java.The JDK also comes with a complete JRE (Java Runtime Environment), so no need of downloading it.

2. Download Eclipse

After setting up the JDK, download the Eclipse and ensure it meets the system requirements.

Eclipse is an IDE (Integrated Development Environment). Eclipse is an open development platform consisting of frameworks, tools and runtimes for building, deploying and managing software.

3. Downloading ADT Plug-in through Eclipse

ADT (Android Development Tool) is a Plug-in for Eclipse that provides tools and features to

Develop the android application quickly.

Steps to setup ADT Plug-in in Eclipse:-

(i) Open Eclipse go to Help => Install New Software . A dialog box will appear.

(ii) Click on Add button. In Add Repository, write any suitable name ('android' or your name) & in Location paste the URL, http://dl-ssl.google.com/android/eclipse.

then click on OK.

(iii) After this, available software by the name "Developer Tools" will appear. Click on this, a list of tools will pop-down, select Android Development Tools and click on Next. Accept the Terms & Conditions and click on Finish. The software starts installing.

CHAPTER 4

4.0Implementation

The steps taken to develop the project are:-

- Install jre then install eclipse i.e.(eclipse-SDK-3.6.2-win32) after that install jdk (cnet2_jdk-6-windows-i586_zip) .
- Install android tools (SDK manager and AVD manager) in ECLIPSE.
- Now,Eclipse provided the feature to develop android projects.
- First we make a list in which all the categories of foods i.e. North Indian food, south Indian foods etc. will be visible.
- In the item of the list we make one image view to show the image of that particular food item and text view to show the name of that item.
- As user click on the item next screen would be visible showing them the item of the selected category food.
- In this user can see the check box also which he can check if he want that food or can uncheck if he do not want that item.
- In these screens we use the android inbuilt **setListAdpater** class to set the image view, text view and check box into the list's items.
- We had copy all the images that we want to use in the **drawable** folder of our android project, and use them with their names.
- To set the list item i.e. Image View, Text View and Check Box we use the android inbuilt **Inflator** class to inflate the layout onto the list item.

4.1Classes Details

User Side Application

```
importandroid.app.Activity;
importandroid.content.Intent;
importandroid.graphics.Bitmap;
importandroid.graphics.BitmapFactory;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.TextView
```

Appetizers.java

```
package com.naman.sample;


import java.util.ArrayList;


import android.R.string;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ExpandableListView;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.ListAdapter;

import android.widget.ListView;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;


import com.naman.sample.SampleActivity.MyThumbnaildapter;


public class Appetizers extends ListActivity implements OnClickListener{

        String  bud_arr[]={"Vegetarian Pakora","Palak Pakoras","Chiken
Pakoras","Samosa Vegetable","Papadams"};

        Button checkout,back;

        final Appetizers second = this;

        int count;

        CheckBox check;

        TextView textfilePath;

        Spinner spin;

        TextView price;

        ImageView imageThumbnail;

        ArrayList<Boolean> itemChecked = new ArrayList<Boolean>();;

        String item_price []={"Rs:","Rs:", "Rs:","Rs:","Rs:"};

        int prices [] = { 190,60,85,80,70, 0};

        String Quantity[]={"1","2","3","4","5","6","7"};

         private Integer[] imgid = {
```

```
R.drawable.appetizers,R.drawable.palakpakora,R.drawable.chickenpakora,R.drawable.s
amosa,R.drawable.papad};

        public class MyThumbnaildapter extends ArrayAdapter<String>{


                private final Context context;

                private final String[] values;


                public MyThumbnaildapter(Context context, int textViewResourceId,

                 String[] objects) {

                 super(context, textViewResourceId, objects);

                this.context= context;

                this.values= objects;

                count=this.getCount();

                for (int i = 0; i < this.getCount(); i++) {

               itemChecked.add(i, false);

                }

                 // TODO Auto-generated constructor stub

                }




                @Override

                public View getView(final int position, final View convertView, ViewGroup
parent) {
```

```
        // TODO Auto-generated method stub

        View row = convertView;
        if(CheckUnchecked.appetizers_veg==1)
    {
        itemChecked.set(0, true);
    }
         if(CheckUnchecked.appetizers_palak==1)
    {
        itemChecked.set(1, true);
    }
         if(CheckUnchecked.appetizers_chicken==1)
    {
        itemChecked.set(2, true);
    }
         if(CheckUnchecked.appetizers_samosa==1)
    {
        itemChecked.set(3, true);
    }
         if(CheckUnchecked.appetizers_papad==1)
    {
        itemChecked.set(4, true);
    }
```

```
                if(row==null){
                 LayoutInflater inflater=getLayoutInflater();
                 row=inflater.inflate(R.layout.second, parent, false);
                 check =(CheckBox)row.findViewById(R.id.checkBox1);
                 price = (TextView)row.findViewById(R.id.textView1);
                 textfilePath = (TextView)row.findViewById(R.id.ItemName);
                 imageThumbnail = (ImageView)row.findViewById(R.id.item);
                 spin=(Spinner)row.findViewById(R.id.spinner1);
                 spin.setPrompt("Quantity");
                }
               // ListView listview = getListView();
               // LayoutInflater layinflate = getLayoutInflater();
                //listview.addHeaderView(layinflate.inflate(R.layout.placeorder, null),
null,false);
      ArrayAdapter<String> adapter = new
ArrayAdapter<String>(getApplicationContext(),
android.R.layout.simple_spinner_item,Quantity);

adapter.setDropDownViewResource(android.R.layout.simple_dropdown_item_1line);
                spin.setAdapter(adapter);
                check.setOnCheckedChangeListener(new OnCheckedChangeListener() {

              public void onCheckedChanged(CompoundButton buttonView,
                  boolean isChecked) {

                if (isChecked && position==0) {
```

```
        itemChecked.set(position, true);
        if(CheckUnchecked.appetizers_veg!=1){
    //  Data.order_name[Data.k] = bud_arr[position];
    // Data.k++;
     Data.Grand_total = Data.Grand_total + prices[position];
     Data.order_place = Data.order_place + "\n" + bud_arr[position];
        }
     CheckUnchecked.appetizers_veg=1;
     // do some operations here
  } else if(position==0 && isChecked==false){
        Log.i("TAG", "unchk");
     itemChecked.set(position, false);
     Data.Grand_total = Data.Grand_total - prices[position];
     CheckUnchecked.appetizers_veg=2;
     // do some operations here
  }
  if (isChecked && position==1) {
        itemChecked.set(position, true);
        if(CheckUnchecked.appetizers_palak!=1){
   //  Data.order_name[Data.k] = bud_arr[position];
   // Data.k++;
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
       }
    CheckUnchecked.appetizers_palak=1;
```

```
    // do some operations here
  } else if(position==1 && isChecked==false){
        Log.i("TAG", "unchk");
    itemChecked.set(position, false);
    Data.Grand_total = Data.Grand_total - prices[position];
    CheckUnchecked.appetizers_palak=2;

    // do some operations here
  }
 if (isChecked && position==2) {
        itemChecked.set(position, true);
        if(CheckUnchecked.appetizers_chicken!=1){
  //  Data.order_name[Data.k] = bud_arr[position];
   // Data.k++;
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
      }
    CheckUnchecked.appetizers_chicken=1;
    // do some operations here
 } else if(position==2 && isChecked==false){
      Log.i("TAG", "unchk");
    itemChecked.set(position, false);
    Data.Grand_total = Data.Grand_total - prices[position];
    CheckUnchecked.appetizers_chicken=2;
```

```
    // do some operations here
}
if (isChecked && position==3) {
        itemChecked.set(position, true);
        if(CheckUnchecked.appetizers_samosa!=1){
  //  Data.order_name[Data.k] = bud_arr[position];
   // Data.k++;
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
       }
    CheckUnchecked.appetizers_samosa=1;
    // do some operations here
} else if(position==3 && isChecked==false){
        Log.i("TAG", "unchk");
    itemChecked.set(position, false);
    Data.Grand_total = Data.Grand_total - prices[position];
    CheckUnchecked.appetizers_samosa=2;
}
if (isChecked && position==4) {
        itemChecked.set(position, true);
        if(CheckUnchecked.appetizers_papad!=1){
  //  Data.order_name[Data.k] = bud_arr[position];
   // Data.k++;
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
```

```
        }
    CheckUnchecked.appetizers_papad=1;
    // do some operations here
 } else if(position==4 && isChecked==false){
        Log.i("TAG", "unchk");
    itemChecked.set(position, false);
    Data.Grand_total = Data.Grand_total - prices[position];
    CheckUnchecked.appetizers_papad=2;
 }
 /*
 if (isChecked && position==1) {
    itemChecked.set(position, true);
   // Data.order_name[Data.k] = bud_arr[position];
   // Data.k++;
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
    // do some operations here
 }
 else {
    itemChecked.set(position, false);
  //  Data.Grand_total = Data.Grand_total - prices[position];
    // do some operations here
 }
 if (isChecked && position==2) {
    itemChecked.set(position, true);
```

```
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
    // do some operations here
}
else {
    itemChecked.set(position, false);
   // Data.Grand_total = Data.Grand_total - prices[position];
    // do some operations here
}
if (isChecked && position==3) {
    itemChecked.set(position, true);
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
    // do some operations here
}
else {
    itemChecked.set(position, false);
    //Data.Grand_total = Data.Grand_total - prices[position];
    // do some operations here
}
if (isChecked && position==4) {
    itemChecked.set(position, true);
    Data.Grand_total = Data.Grand_total + prices[position];
    Data.order_place = Data.order_place + "\n" + bud_arr[position];
    // do some operations here
```

```
        }
        else {
          itemChecked.set(position, false);
          // do some operations here
        } */

      }
    });
        checkout.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent intent = new Intent ();
intent.setClass(second, Place_Order.class);
startActivity(intent);
finish();
            }
      });
       back.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub
                if(v.getId()==R.id.backbtn){
                    onBackPressed();
                finish();
```

```
                    }
                }
            });
             check.setChecked(itemChecked.get(position));
             price.setText(item_price[position] + " " +
Integer.toString(prices[position]));
             price.setId(position);

             textfilePath.setText(bud_arr[position]);
             textfilePath.setId(position);

     imageThumbnail.setImageResource(imgid[position]);
    // Button checkout = new Button(getApplicationContext());
    // checkout.setGravity(0x50);

             return row;
            }

        }
      protected void onListItemClick(ListView l, View v, int position, long id) {

            ViewGroup row = (ViewGroup)l.getChildAt(position);
```

```
            if(position==0)
            {

                //check.setChecked(true);
                Toast.makeText(getApplicationContext(), "north",
Toast.LENGTH_SHORT).show();
            }
            //int check_id=check.getId();



        }


 @Override
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.mains);
   checkout=(Button)findViewById(R.id.checkout);
   back=(Button)findViewById(R.id.backbtn);
   //MyThumbnaildapter;
  Data.backbtnclass=1;
   ListView l = (ListView)findViewById(android.R.id.list);
   l.setAdapter(new MyThumbnaildapter(Appetizers.this, R.layout.second, bud_arr));
}
```

```
public void onClick(View v) {
        // TODO Auto-generated method stub


}
}
```

PlaceOrder.java

```
package com.naman.sample;
import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.telephony.SmsManager;
import android.text.Editable;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class Place_Order extends Activity{
        SharedPreferences prefs;
        SharedPreferences.Editor editor;
```

```
Button order;
TextView c_name,grand_total;
String c_address;
EditText additional_instruction,address;
String customer_order;
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.placeorder);

        Context ctx= getApplicationContext();
prefs =PreferenceManager.getDefaultSharedPreferences(ctx);
editor =prefs.edit();

        order=(Button)findViewById(R.id.button1);
        additional_instruction=(EditText)findViewById(R.id.editText2);
        address=(EditText)findViewById(R.id.editText1);
        c_name = (TextView)findViewById(R.id.textname);
        grand_total=(TextView)findViewById(R.id.textView7);
        //order = (Button) findViewById(R.id.btnSendSMS);
        grand_total.setText(String.valueOf(Data.Grand_total));
        final String n = prefs.getString("names", "");
        final String a = prefs.getString("addresss", "");
        final String p = prefs.getString("numbers", "");
```

```
            Log.i("TAG", "hi");
            c_name.setText(n);
            address.setText(a);

    order.setOnClickListener(new View.OnClickListener()
    {
        public void onClick(View v)
        {
        // String phone = String.valueOf(phone_number.getText());
        // customer_order ;
        String add_instruiction = String.valueOf(additional_instruction.getText());
                String message = "Customer name :" + n + "\n Address : " + a + "\n Phone
No." + p + "Order: " + Data.order_place + "\n Additional Instructions :" +
add_instruiction + "\n Grand Total : Rs."+Data.Grand_total;
          sendSMS("9897931923", message);
        }
    });
        }
        private void sendSMS(String phoneNumber, String message)
         {
            SmsManager sms = SmsManager.getDefault();
            sms.sendTextMessage(phoneNumber, null, message, null, null);
         }

}
```

Restaurant Side Application

MainMenu.java

```
package com.example.rec;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainMenu extends Activity implements OnClickListener {
Button msg,update;
Intent i;
@Override
 public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main_menu);

msg=(Button)findViewById(R.id.button1);

update=(Button)findViewById(R.id.button2);

msg.setOnClickListener(this);

update.setOnClickListener(this);


}
```

```
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
if(v==msg)
{
i=new Intent(this,Sms.class);
startActivity(i);
finish();
if(v==update)
{
i=new Intent(this,Updateprices.class);
startActivity(i);
finish();
}
}
}
```

SmsReceiveActivity.java

```
package com.example.rec;

import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
```

```
import android.widget.Toast;


public class SmsrecActivity extends BroadcastReceiver{

   /** Called when the activity is first created. */

private static final String SMS_RECEIVED =
"android.provider.Telephony.SMS_RECEIVED";

 static SmsMessage[] messages;

 static int c=0;

@Override

public void onReceive(Context context, Intent intent) {

// TODO Auto-generated method stub

if (intent.getAction().equals(SMS_RECEIVED)) {

 Bundle bundle = intent.getExtras();

        if (bundle != null) {

           Object[] pdus = (Object[])bundle.get("pdus");

            messages = new SmsMessage[pdus.length];

           for (int i = 0; i < pdus.length; i++) {

              messages[i] = SmsMessage.createFromPdu((byte[])pdus[i]);

           }

           if (messages.length > -1) {

  Intent i=new Intent(context,Sms.class);


i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

 c++;

 context.startActivity(i);

Toast.makeText(context, "Message recieved: " + messages[0].getMessageBody(),
7000).show();
```

```
 }
 }
 }
}
}
```

UpdatePrices.java

```
package com.example.rec;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
```

```
import org.json.JSONObject;


import android.app.Activity;
import android.app.AlertDialog;
import android.content.ContentValues;
import android.content.Context;
import android.content.DialogInterface;
import android.database.Cursor;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.StrictMode;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;


public class Updateprices extends Activity{
```

```
static String[] items;
String[] prices;
static String[] item_value=new String[100];
static String[] item_name=new String[100];


ListView lv;
CheckBox cb;
Button update,sync;
String uvalue;
static int count=0;
static TextView[] updated=new TextView[100];
public void create(String price, final TextView tv1,final String item,final int position)
{
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Update Prices");
builder.setMessage("Current Price "+price);

  // Use an EditText view to get user input.
  final EditText input = new EditText(this);
  builder.setView(input);
  builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
  @Override
    public void onClick(DialogInterface dialog, int whichButton) {
       uvalue = input.getText().toString();
       tv1.setText(uvalue);
```

```
item_value[count]=uvalue;
prices[position]=uvalue;
item_name[count]=item;
Log.i("TAG",uvalue);
DBAdapter dbAdapter =
DBAdapter.getDBAdapterInstance(getApplicationContext());
try {
dbAdapter.createDataBase();
} catch (IOException e) {
}
dbAdapter.openDataBase();
try {
String query = "UPDATE Prices SET Price="+item_value[count]+" where
Item_Name='"+item_name[count]+"'";
Log.i("TAG","Query executed");
Cursor c = dbAdapter.selectRecordsFromDB(query, null);
ContentValues cv= new ContentValues();
cv.put("Price", item_value[count]);
boolean c1=dbAdapter.updateRecordInDB("Prices", cv,
"Item_Name='"+item_name[count]+"'", null);
count++;
}
catch (Exception e) {
e.printStackTrace();
}
dbAdapter.close();
```

```
    }
});

builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {

    }
    });
builder.show();                                                    }

static class ViewHolder
{
TextView item,price;
Button cprice;
}
public class MyThumbnaildapter extends ArrayAdapter<String>{
public MyThumbnaildapter(Context context, int textViewResourceId,

String[] objects) {

super(context, textViewResourceId, objects);
 // TODO Auto-generated constructor stub
 }
 @Override
 public View getView(final int position, View convertView, ViewGroup parent) {
```

```
ViewHolder holder = null;
View row = convertView;
LayoutInflater inflater=getLayoutInflater();
if(row==null){
row=inflater.inflate(R.layout.prices_list, parent, false);

holder = new ViewHolder();

holder.item=(TextView)row.findViewById(R.id.textView1);

holder.price=(TextView)row.findViewById(R.id.textView2);

holder.cprice=(Button)row.findViewById(R.id.button1);

row.setTag (holder);
}
else
{
// We recycle a View that already exists

holder = (ViewHolder)row.getTag ();

holder.price.setText(prices[position]);

holder.cprice.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
ViewHolder vh=new ViewHolder();
```

```
vh.item=(TextView)rl.findViewById(R.id.textView1);

vh.price=(TextView)rl.findViewById(R.id.textView2);

create(vh.price.getText().toString(),vh.price,vh.item.getText().toString(),position);
}
});
return row;
}
}
@Override
public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.prices);

lv=(ListView)findViewById(R.id.listView1);

update=(Button)findViewById(R.id.button1);

sync=(Button)findViewById(R.id.button2);

StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);

sync.setOnClickListener(new OnClickListener() {
@Override
```

```
public void onClick(View v) {
// TODO Auto-generated method stub
AsyncTask task = new ProgressTask1(Updateprices.this).execute();
}
});

DBAdapter dbAdapter = DBAdapter.getDBAdapterInstance(this);
try {
dbAdapter.createDataBase();
} catch (IOException e) {
}
dbAdapter.openDataBase();
 try {
String query = "SELECT * FROM Prices";
Cursor c = dbAdapter.selectRecordsFromDB(query, null);
items=new String[c.getCount()];
 prices=new String[c.getCount()];
int price = c.getColumnIndex("Price");
int item=c.getColumnIndex("Item_Name");
String item1;
double price1;
int i=0;
c.moveToFirst();
if (c != null) {
do {
price1=c.getDouble(price);
```

```
item1=c.getString(item);
items[i]=item1;
prices[i++]=Double.toString(price1);
}while(c.moveToNext());
 }catch (Exception e) {
e.printStackTrace();
 }
  dbAdapter.close();
 update.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
AsyncTask task = new ProgressTask(Updateprices.this).execute();
});

lv.setAdapter(new MyThumbnaildapter(Updateprices.this, R.layout.prices_list,items));
}
 static void update_server()
 {

HttpClient httpclient = new DefaultHttpClient();

HttpPost httppost = new HttpPost("http://jannatyahan.com/prices.php");

for(int i=0;i<count;i++)
{

String Query="UPDATE Prices Set Price="+item_value[i]+ " where
Item_Name='"+item_name[i]+"'";
```

```
try {
        // Add user name and password
List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(1);
 nameValuePairs.add(new BasicNameValuePair("Query", Query));
 httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
   HttpResponse response = httpclient.execute(httppost);
String str = inputStreamToString(response.getEntity().getContent()).toString();
  } catch (ClientProtocolException e) {
     e.printStackTrace();
     } catch (IOException e) {
      e.printStackTrace();
     }
}
}
private static StringBuilder inputStreamToString(InputStream is) {
   String line = "";
   StringBuilder total = new StringBuilder();
   // Wrap a BufferedReader around the InputStream
   BufferedReader rd = new BufferedReader(new InputStreamReader(is));
   // Read response until the end
   try {
   while ((line = rd.readLine()) != null) {
     total.append(line);
   }
   } catch (IOException e) {
    e.printStackTrace();
```

```
    }
    // Return full string
    return total;
   }
 void sync_rest()
    {
    String result ="";
 String Query="SELECT * FROM Prices";
  InputStream is = null;
   ArrayList<NameValuePair> nameValuePairs1 = new ArrayList<NameValuePair>(1);

nameValuePairs1.add(new BasicNameValuePair("Query",Query));
  try{

HttpClient httpclient = new DefaultHttpClient();

HttpPost httppost = new HttpPost("http://jannatyahan.com/prices1.php");

httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs1));

HttpResponse response = httpclient.execute(httppost);

HttpEntity entity = response.getEntity();

is = entity.getContent();
   }catch(Exception e){
  }
   //convert response to string
   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();
 result=sb.toString();
 result.trim();
  }catch(Exception e){
Log.e("log_tag", "Error converting result "+e.toString());
  }

//parse json data

String returnString = null;
try{


JSONArray jArreglo = new JSONArray(result);

for(int i=0 ; i<jArreglo.length(); i++)

{
```

```
JSONObject json_data = jArreglo.getJSONObject(i);

items[i]=json_data.getString("Item_Name");

prices[i]= json_data.getString("Price");

DBAdapter dbAdapter = DBAdapter.getDBAdapterInstance(getApplicationContext());

try {
dbAdapter.createDataBase();
} catch (IOException e) {
}
dbAdapter.openDataBase();

try {

String query = "UPDATE Prices SET Price="+prices[i]+" where
Item_Name='"+items[i]+"'";
ContentValues cv= new ContentValues();
cv.put("Price", prices[i]);
 boolean c1=dbAdapter.updateRecordInDB("Prices", cv, "Item_Name='"+items[i]+"'",
null);
}
 catch (Exception e) {
e.printStackTrace();
      }
      dbAdapter.close();
     }
```

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

DBAdapter.java

```
package com.example.rec;



import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;

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

```
public class DBAdapter extends SQLiteOpenHelper {

    private static String DB_PATH = "";
    private static final String DB_NAME = "prices1.sqlite";
    private SQLiteDatabase myDataBase;
    private final Context myContext;

    private static DBAdapter mDBConnection;

    /**
     * Constructor Takes and keeps a reference of the passed context in order to
     * access to the application assets and resources.
     *
     * @param context
     */
    private DBAdapter(Context context) {
        super(context, DB_NAME, null, 1);
        this.myContext = context;
//      Log.i("TAG", "DMNAME---> " + DB_NAME);
        DB_PATH = "/data/data/"
                + context.getApplicationContext().getPackageName()
                + "/databases/";
        // The Android's default system path of your application database is
        // "/data/data/mypackagename/databases/"
```

```
}

/**
 * getting Instance
 *
 * @param context
 * @return DBAdapter
 */
public static synchronized DBAdapter getDBAdapterInstance(Context context) {
        if (mDBConnection == null) {
                mDBConnection = new DBAdapter(context);
        }
        return mDBConnection;
}

/**
 * Creates an empty database on the system and rewrites it with your own
 * database.
 **/
public void createDataBase() throws IOException {
        boolean dbExist = checkDataBase();
        SQLiteDatabase db_Read = null;
        if (dbExist) {
                // do nothing - database already exist
```

```
        } else {
                // By calling following method
                // 1) an empty database will be created into the default system path
                // of your application
                // 2) than we overwrite that database with our database.
                db_Read = this.getReadableDatabase();
                db_Read.close();
                try {
                        copyDataBase();
                } catch (IOException e) {
                        throw new Error("Error copying database");
                }
        }
}

/**
 * Check if the database already exist to avoid re-copying the file each
 * time you open the application.
 *
 * @return true if it exists, false if it doesn't
 */
private boolean checkDataBase() {
        SQLiteDatabase checkDB = null;
        try {
```

```
            String myPath = DB_PATH + DB_NAME;
            checkDB = SQLiteDatabase.openDatabase(myPath, null,
                        SQLiteDatabase.OPEN_READONLY);

        } catch (SQLiteException e) {
            // database does't exist yet.
        }
        if (checkDB != null) {
            checkDB.close();
        }
        return checkDB != null ? true : false;
}

/**
 * Copies your database from your local assets-folder to the just created
 * empty database in the system folder, from where it can be accessed and
 * handled. This is done by transfering bytestream.
 * */
private void copyDataBase() throws IOException {
        // Open your local db as the input stream
        InputStream myInput = myContext.getAssets().open(DB_NAME);
        // Path to the just created empty db
        String outFileName = DB_PATH + DB_NAME;
        // Open the empty db as the output stream
```

```
        OutputStream myOutput = new FileOutputStream(outFileName);
        // transfer bytes from the inputfile to the outputfile
        // byte[] buffer = new byte[1024];
        byte[] buffer = new byte[3000];
        int length;
        while ((length = myInput.read(buffer)) > 0) {
                myOutput.write(buffer, 0, length);
        }
        // Close the streams
        myOutput.flush();
        myOutput.close();
        myInput.close();
}

/**
 * Open the database
 *
 * @throws SQLException
 */
public void openDataBase() throws SQLException {
        String myPath = DB_PATH + DB_NAME;
        myDataBase = SQLiteDatabase.openDatabase(myPath, null,
                        SQLiteDatabase.OPEN_READWRITE);

}
```

```
/**
 * Close the database if exist
 */
@Override
public synchronized void close() {
        if (myDataBase != null)
                myDataBase.close();
        super.close();
}

/**
 * Call on creating data base for example for creating tables at run time
 */
@Override
public void onCreate(SQLiteDatabase db) {
}

/**
 * can used for drop tables then call onCreate(db) function to create tables
 * again - upgrade
 */
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
}

// ---------- CRUD Functions ----------

/**
 * This function used to select the records from DB.
 *
 * @param tableName
 * @param tableColumns
 * @param whereClase
 * @param whereArgs
 * @param groupBy
 * @param having
 * @param orderBy
 * @return A Cursor object, which is positioned before the first entry.
 */
public Cursor selectRecordsFromDB(String tableName, String[] tableColumns,
                String whereClase, String whereArgs[], String groupBy,
                String having, String orderBy) {
        return myDataBase.query(tableName, tableColumns, whereClase,
whereArgs,
                        groupBy, having, orderBy);

}
```

```
/**
 * select records from db and return in list
 *
 * @param tableName
 * @param tableColumns
 * @param whereClase
 * @param whereArgs
 * @param groupBy
 * @param having
 * @param orderBy
 * @return ArrayList<ArrayList<String>>
 */
public ArrayList<ArrayList<String>> selectRecordsFromDBList(
        String tableName, String[] tableColumns, String whereClase,
        String whereArgs[], String groupBy, String having, String orderBy)
{

    ArrayList<ArrayList<String>> retList = new
ArrayList<ArrayList<String>>();
    ArrayList<String> list = new ArrayList<String>();
    Cursor cursor = myDataBase.query(tableName, tableColumns, whereClase,
            whereArgs, groupBy, having, orderBy);
    if (cursor.moveToFirst()) {
        do {
            list = new ArrayList<String>();
```

```
                for (int i = 0; i < cursor.getColumnCount(); i++) {
                        list.add(cursor.getString(i));
                }
                retList.add(list);
            } while (cursor.moveToNext());
        }
        if (cursor != null && !cursor.isClosed()) {
            cursor.close();
        }
        return retList;

}

/**
 * This function used to insert the Record in DB.
 *
 * @param tableName
 * @param nullColumnHack
 * @param initialValues
 * @return the row ID of the newly inserted row, or -1 if an error occurred
 */
public long insertRecordsInDB(String tableName, String nullColumnHack,
            ContentValues initialValues) {
        return myDataBase.insert(tableName, nullColumnHack, initialValues);
```

```
    }

    /**
     * This function used to update the Record in DB.
     *
     * @param tableName
     * @param initialValues
     * @param whereClause
     * @param whereArgs
     * @return true / false on updating one or more records
     */
    public boolean updateRecordInDB(String tableName,
                    ContentValues initialValues, String whereClause, String
whereArgs[]) {
            return myDataBase.update(tableName, initialValues, whereClause,
                            whereArgs) > 0;

    }

    /**
     * This function used to update the Record in DB.
     *
     * @param tableName
     * @param initialValues
     * @param whereClause
```

```
* @param whereArgs
* @return 0 in case of failure otherwise return no of row(s) are updated
*/
public int updateRecordsInDB(String tableName, ContentValues initialValues,
                String whereClause, String whereArgs[]) {
        Log.i("TAG","Called Update method ");
        Log.i("TAG","Updated Sucessfully ");
        return myDataBase.update(tableName, initialValues, whereClause,
                        whereArgs);
}

/**
* This function used to delete the Record in DB.
*
* @param tableName
* @param whereClause
* @param whereArgs
* @return 0 in case of failure otherwise return no of row(s) are deleted.
*/
public int deleteRecordInDB(String tableName, String whereClause,
                String[] whereArgs) {
        return myDataBase.delete(tableName, whereClause, whereArgs);
}
```

```
// ---------- Select Raw Query Functions ----------

/**
 * apply raw Query
 *
 * @param query
 * @param selectionArgs
 * @return Cursor
 */
public Cursor selectRecordsFromDB(String query, String[] selectionArgs) {
        return myDataBase.rawQuery(query, selectionArgs);
}

/**
 * apply raw query and return result in list
 *
 * @param query
 * @param selectionArgs
 * @return ArrayList<ArrayList<String>>
 */
public ArrayList<ArrayList<String>> selectRecordsFromDBList(String query,
                String[] selectionArgs) {
        Log.i("TAG", "Executing The Query");
```

```
        ArrayList<ArrayList<String>> retList = new
ArrayList<ArrayList<String>>();
        ArrayList<String> list = new ArrayList<String>();
        Cursor cursor = myDataBase.rawQuery(query, selectionArgs);
        if (cursor.moveToFirst()) {
            do {
                list = new ArrayList<String>();
                for (int i = 0; i < cursor.getColumnCount(); i++) {
                    list.add(cursor.getString(i));
                }
                retList.add(list);
            } while (cursor.moveToNext());
        }
        if (cursor != null && !cursor.isClosed()) {
            cursor.close();
        }
        return retList;
    }

}
```

4.2 Data Flow Diagram

Context Level Diagram:

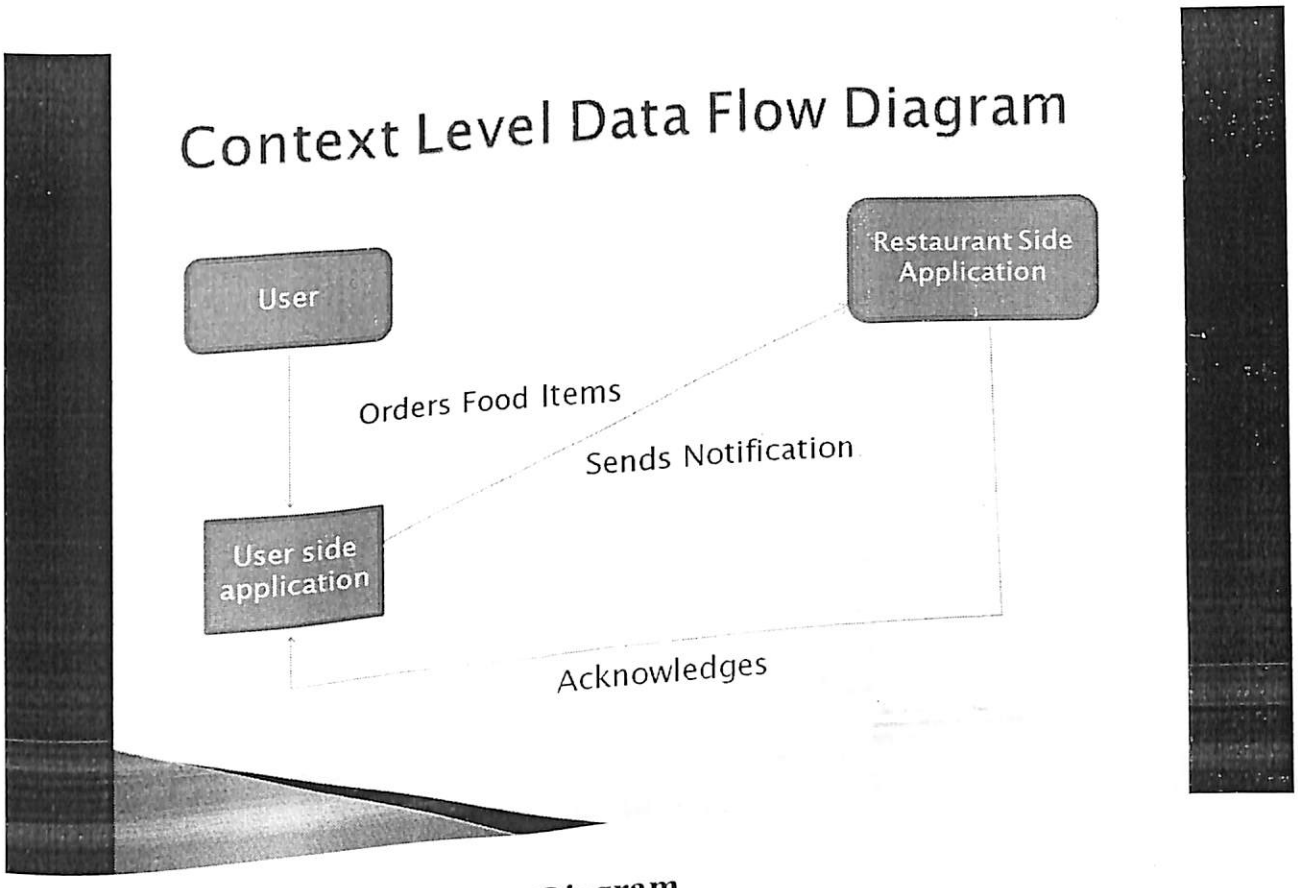

Figure 4:Context Level Data Flow Diagram

4.3Sample Source code(Coding Progress Till Now)

Layout (main.xml)

```
<?xmlversion="1.0"encoding="UTF-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="80dp"
android:background="@drawable/back"
android:orientation="horizontal">

<ImageView
android:id="@+id/Thumbnail"
android:layout_width="80dp"
android:layout_height="80dp"

android:src="@drawable/up"/>
```

```
<TextView
android:id="@+id/FilePath"
android:layout_width="150dp"

android:textColor="#000000"
android:gravity="center_vertical"
android:paddingLeft="10dp"
android:layout_height="80dp"/>


</LinearLayout>
```

Inflate.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="80dp"
android:background="@drawable/back"
android:orientation="horizontal">

<ImageView
android:id="@+id/item"
android:layout_width="80dp"
android:layout_height="80dp"

android:src="@drawable/up"/>

<TextView
android:id="@+id/ItemName"
android:layout_width="150dp"

android:textColor="#000000"
android:gravity="center_vertical"
android:paddingLeft="10dp"
android:layout_height="80dp"/>
<CheckBox
android:id="@+id/checkBox1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
android:focusable="false"
/>

</LinearLayout>
```

AndroidManifest.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="com.naman.sample"
android:versionCode="1"
android:versionName="1.0">
```

```
<uses-sdkandroid:minSdkVersion="8"/>
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission/>

<application
android:icon="@drawable/ic_launcher"
android:label="Sample">
<activity
android:label="Sample"
android:name=".SampleActivity">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>

<categoryandroid:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activityandroid:name="SecondActivity"></activity>
<activityandroid:name="SouthIndian"></activity>
<activityandroid:name="ChineseFoods"></activity>
<activityandroid:name="Italian"></activity>
</application>

</manifest>
```

4.3Output (Screens)

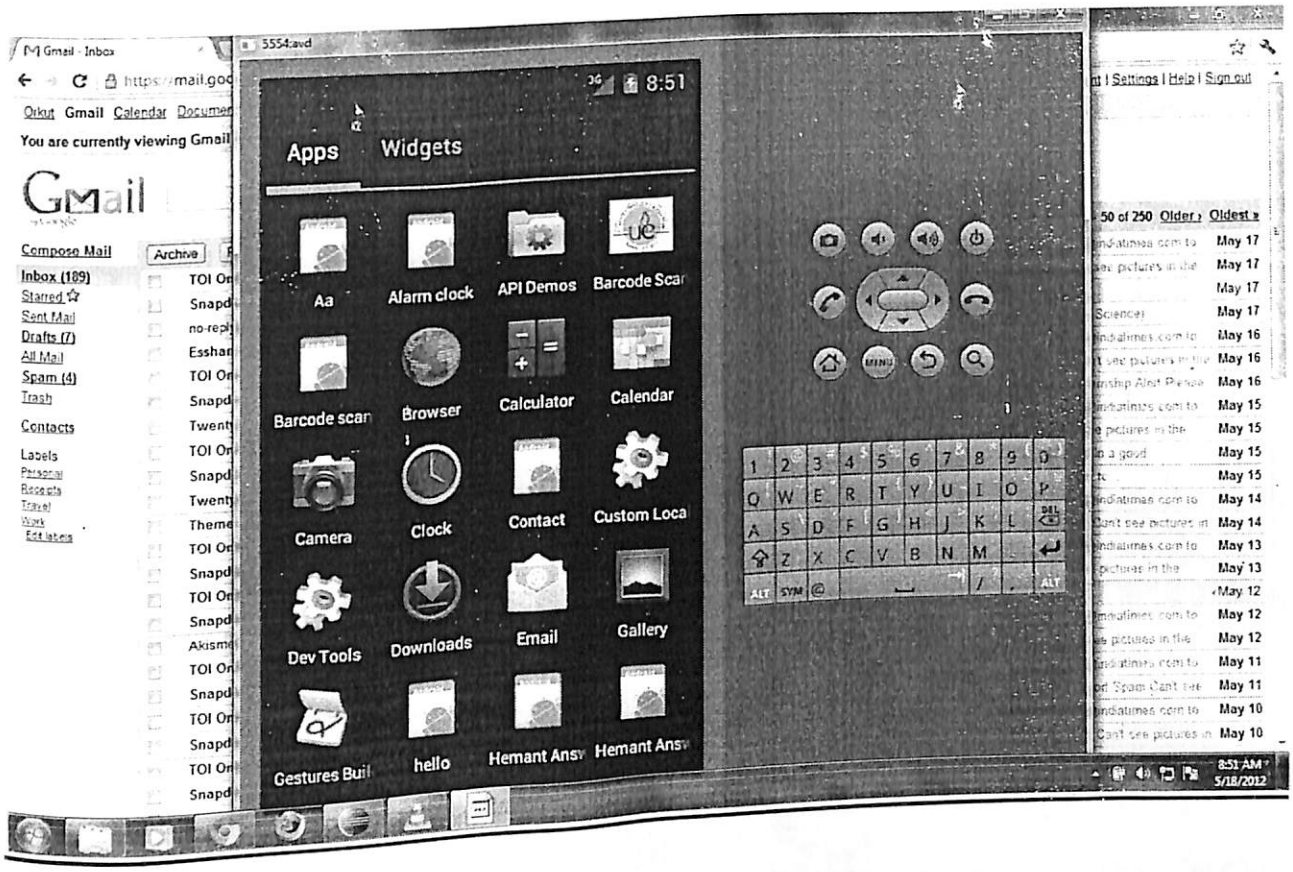

Figure 5:Emulator Menu Screen

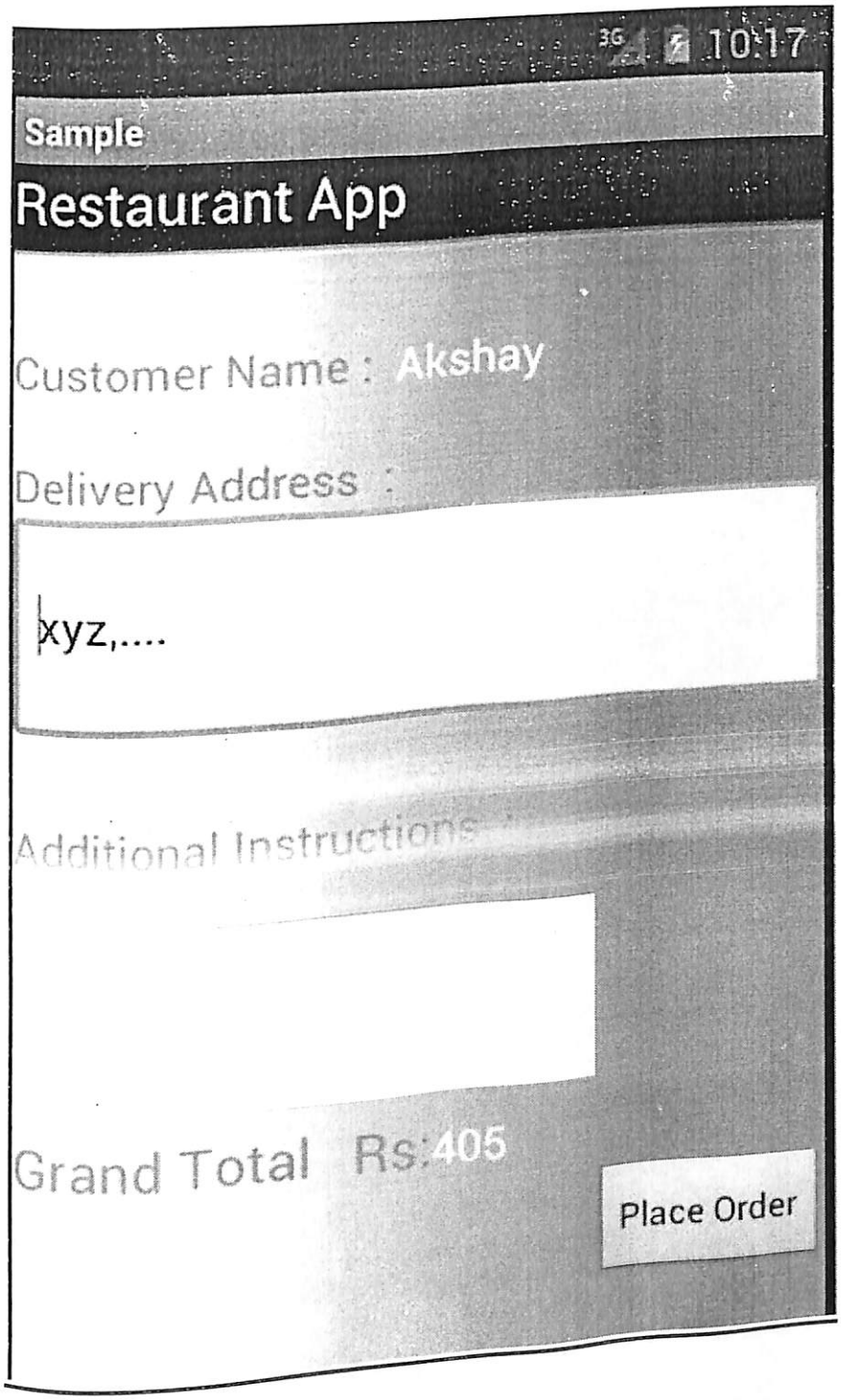

Figure 6:Registration of user

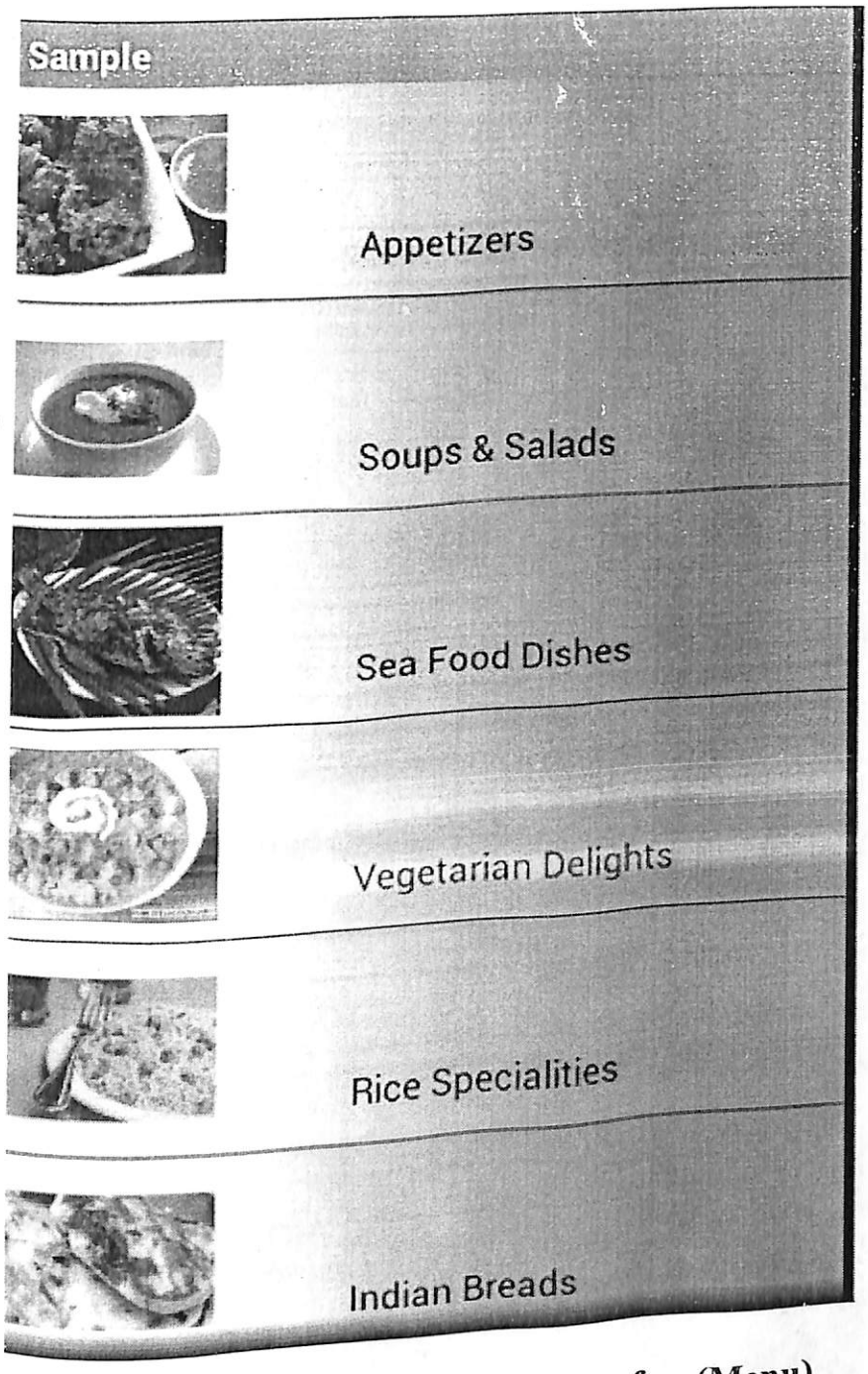

Figure 7:Restaurant Application Interface(Menu)

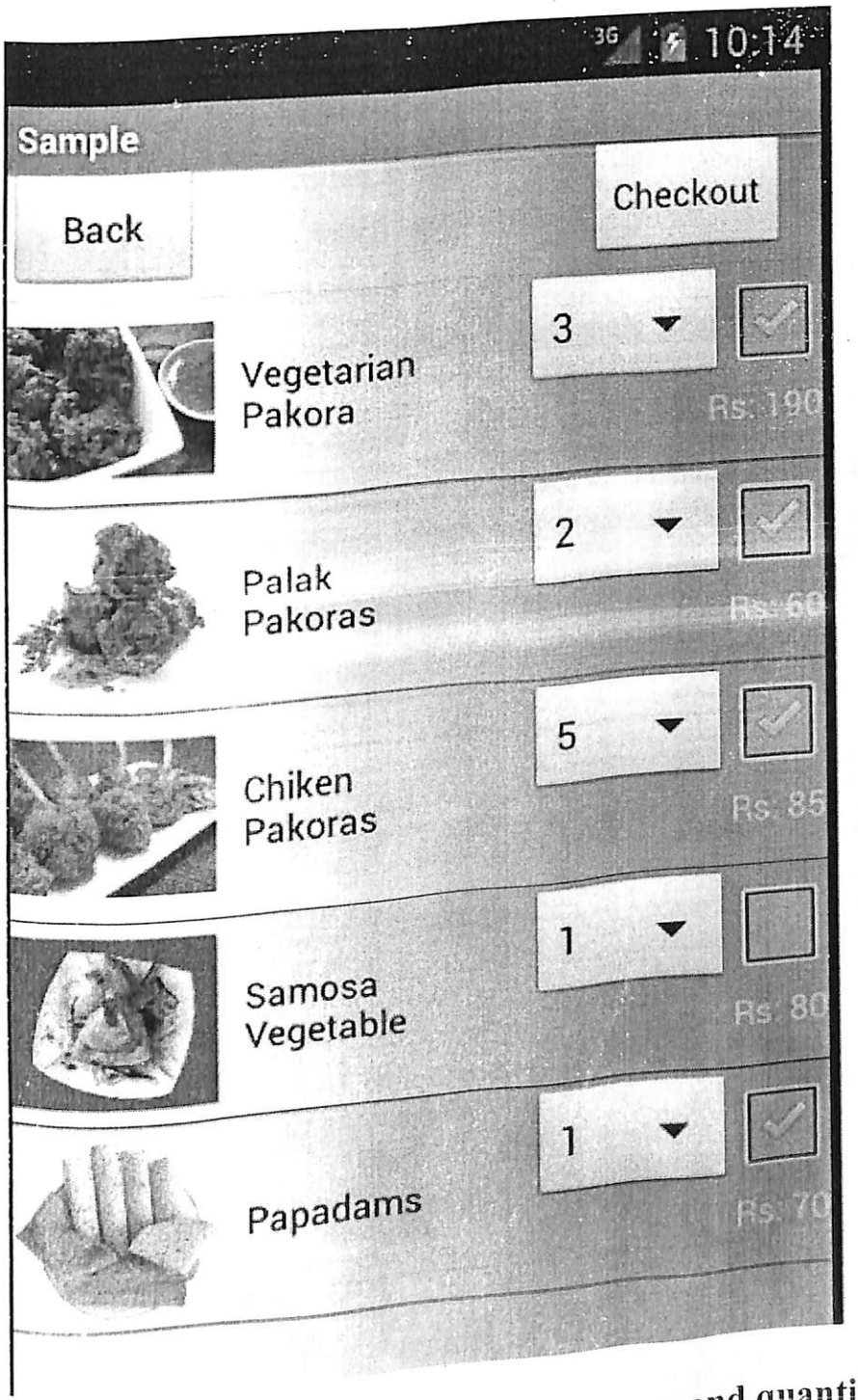

Figure 8:Interface for selecting food items and quantity as desired

Progress Till Now

Till now we have developed the client side application with the main menu available, showing the main restaurant course menu. This main menu contains different food sections like north Indian, south Indian, Chinese, etc. Selecting one of these food section will further show a menu that contains the list of the different types of foods (for e.g.North Indian food contains rice, salads, dals, etc.). From these different food items we can select our desired food item and the quantity.

Restaurant side application receives the order and notifies the user.The restaurant side application is also connected to the server and can update prices of food items as and when desired.

CHAPTER 5

5.0 Testing

5.1 Introduction

Testing is a process of analyzing a system or system components to detect the differences between specified and observed behavior. In other words, testing is a fault detection technique that tries to create failures or erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer.

The application developed can be tested using the android virtual device(AVD) emulator.For testing to be done in android phones the project developed needs to be exported using a keystroke and filling the application sign up form for certification.

While installing the application on android phone a parsing error is likely to occur which is because of high SDK version which is set by default.Therefore,it is necessary to set the min SDK version to 4 so as to avoid Parsing error.In the android manifest file change as shown below:

<uses-sdkandroid:minSdkVersion="4" />

Device Based:

The devices with android operating system were able to run the application successfully in all versions of android i.e. from 1.2 to 4.2(Jelly Bean). And this application will run only on android devices.

CONCLUSION

The android application brings android technology more close to real world. It provides convenient means of ordering food. It is convenient for the restaurant to manage the orders delivery. It also helps promotion of the restaurant.Most importantly,you can have food any time any place.

BIBLIOGRAPHY

http://stackoverflow.com

http://en.wikipedia.org/wiki/Android

http://androidfordev.blogspot.com

http://www.jannatyahan.com/android