**AIML BASED FRIENDLY CONVERSING CHATBOT**

**Contents:**

**1. Introduction**

**1.1. PROBLEM DEFINITION**

**1.2 EXISTING SYSTEM**

**1.3 PROPOSED SYSTEM**

**1.3.1 Modules**

**1.4 SCOPE OF PROJECT**

**2. LITERATURE SURVEY**

**2.1. JAVA**

**2.2. ANDROID**

**2.2.1 About Native code**

**2.2.2 Creating an android project**

**2.2.3 To create an AVD with the AVD manager**

**2.2.4 To Create or Modify a Launch Configuration**

**2.3. SQLITE**

**2.3.1Design**

**2.3.2Features**

**3. SYSTEM ANALYSIS**

**3.1 SOFTWARE REQUIREMENTS**

**3.2 HARDWARE REQUIREMENTS**

### 3.3 FEASIBILITY STUDY

**3.3.1 Economic Feasibility**

**3.3.2 Operational Feasibility**

**3.3.3** **Technical Feasibility**

**4. SYSTEM DESIGN**

**4.1 ARCHITECTURE DIAGRAM**

**4.2 DATA FLOW DIAGRAMS**

**4.3 UNIFIED MODELING LANGUAGE (UML)**

**5. SYSTEM CODING & IMPLEMENTATION**

**5.1 Sample Code**

**5.2 SCREEN SHOTS**

**6. TESTING AND VALIDATION**

**6.1 Unit Testing**

**6.2 Integration Testing**

**6.3 Validation Testing**

**6.4 System Testing**

**6.5 Test cases**

**7. CONCLUSIONS**

**8. REFERENCES**

**INTRODUCTION**

1. **Introduction**

**1.1. PROBLEM DEFINITION:**

Intelligent digital assistants or chatbot is beginning of a new tech era.

A bot is a software application that automates a task. A chatbot automates chat. Chatbots provide a [Conversational User Interface](http://docs.botcentral.ai/glossary/#term-cui), that allows users to interact with your system through a [messaging platform](http://docs.botcentral.ai/glossary/#term-messaging-platform) . The chatbot can receive [input](http://docs.botcentral.ai/glossary/#term-input) from the [user](http://docs.botcentral.ai/glossary/#term-user), perform tasks (place orders, fetch information), and respond to the user with text, media or action. Our chatbot is mainly for the purpose of a friendly conservation with user to make him feel it’s like a friend to you and also perform some tasks like calling and opening the apps .

**1.2 EXISTING SYSTEM:**

# In the existing system the Friendly conversing chatbots are generally used for the conversation with bot which makes us feel that we are not alone. And also As in the existing system in many Corporate Companies, Telecom Sectors we have still usage of huge man power where some times human can make mistakes as it’s a natural tendency and keeping to attain perfect solution we have seen evolution in Technology that’s the reason we are paving our way towards Artificial Intelligence . So these chatbots are used to get the automated answers.

**1.3 PROPOSED SYSTEM:**

The dynamic intelligence of chatbots will allow them to converse with users as in a way we converse and communicate in real-life situations. Conversational communication skill of the

chatbot will not only make it a trendy and promising technology but also empower them to deliver what we are looking for response in human terms and also our virtual assisting chatbot is used for making actions like calling and opening applications by passing the commands in form of conversation.

**1.4 SCOPE OF PROJECT:**

A chatbot is a conversational agent that interacts with users using natural language. Multi chatbots are available to serve in different domains. However, the knowledge base of chatbots is hand coded in its brain. This paper

presents an overview of ALICE chatbot, its AIML format, and our experiments to generate different prototypes of ALICE automatically based on a corpus approach. A description of developed software which converts readable text (corpus) into AIML format is presented alongside with describing the different corpora we used. Our trials revealed the possibility of generating useful prototypes without the need for sophisticated natural language processing or complex machine learning techniques. These prototypes were used as tools to practice different languages, to visualize corpus, and to provide answers for questions.

Keywords. Chatbot, ALICE, AIML, corpus,

machine learning.

.

# 

**LITERATURE SURVEY**

1. **LITERATURE SURVEY**

**2.1. JAVA**

Java is an object-oriented language, and is very similar to C++. Java is simplified to eliminate language features that cause common programming errors. Java source code files are compiled into a format called byte code, which can then be executed by a Java interpreter.Features being

1. **Platform Independent**

The programs written on one platform can run on any platform provided the platform must have the JVM.

1. **Portable**

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM.

1. **Simple**

Programs are easy to write and debug because java does not use the pointers explicitly. It also has the automatic memory allocation and deallaocation system.

1. **Multithreaded**

Multithreading means a single program having different threads executing independently at the same time.

1. **Robust**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages.

1. **Object Oriented**

To be an Object Oriented language, any language must follow at least the four characteristics.

* Inheritance
* Encapsulation
* Polymorphism
* Dynamic binding

1. **Distributed**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access to the files from any remote machine on the internet rather than writing codes on their local system.

1. **Secure**

All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk.

1. **High Performance**

In the beginning interpretation of byte code resulted in slow performance but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

1. **Integrated**

Java is an interpreted language as well. Programs run directly from the source code.

**2.2. ANDROID**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android SDK includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include x86-architecture computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

**2.2.1 About Native code**:

Libraries written in C and other languages can be compiled to ARM native code and installed, but the Native Development Kit is not yet officially supported by Google. Native classes can be called from Java code running under the Dalvik VM using the System. Load Library call, which is part of the standard Android Java classes.

**2.2.2 Creating an android project**

The ADT plug-in provides a New Project Wizard that you can use to quickly create a new Android project (or a project from existing code). To create a new project:

* Select File > New > Project.
* Select Android > Android Project, and click Next.
* Select the contents for the project:
* Enter a Project Name. This will be the name of the folder where your project is created.
* Under Contents, select Create new project in workspace. Select your project workspace location.
* Under Target, select an Android target to be used as the project's Build Target. The Build Target specifies which Android platform you'd like your application built against.
* Unless you know that you'll be using new APIs introduced in the latest SDK, you should select a target with the lowest platform version possible, such as Android 1.1.
* Under Properties, fill in all necessary fields :

Enter an Application name. This is the human-readable title for your application — the name that will appear on the Android device.

1. Enter a Package name. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.
2. Select Create Activity (optional, of course, but common) and enter a name for your main Activity class.
3. Enter a Min SDK Version. This is an integer that indicates the minimum API Level required to properly run your application. Entering this here automatically sets the min Sdk Version attribute in the [<uses-sdk>](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html) of your Android Manifest file. If you're unsure of the appropriate API Level to use, copy the API Level listed for the Build Target you selected in the Target tab.
4. Click Finish.

**2.2.3 To create an AVD with the AVD manager:**

* Select Window > Android SDK and AVD Manager, or click the Android SDK and AVD Manager icon (a black device) in the Eclipse toolbar.
* In the Virtual Devices panel, you'll see a list of existing AVDs. Click New to create a new AVD.
* Fill in the details for the AVD.
* Give it a name, a platform target, an SD card image (optional), and a skin (HVGA is default).
* Click Create AVD.

When you first run a project as an Android Application, ADT will automatically create a run configuration. The default run configuration will launch the default project Activity and use automatic target mode for device selection (with no preferred AVD).

**2.2.4 To Create or Modify a Launch Configuration**

Follow these steps as appropriate for your Eclipse version:

* Open the run configuration manager.
* In Eclipse 3.3 ,select **Run > Open Run Dialog** (or Open Debug Dialog)
* In Eclipse 3.4 (Ganymede), select **Run > Run Configurations** (or Debug Configurations)
* Expand the Android Application item and create a new configuration or open an existing one.

**2.3. SQLITE**

**SQLite** is an ACID-compliant [embedded](http://en.wikipedia.org/wiki/Embedded_database) [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) contained in a relatively small C programming library. The source code for SQLite is in the public domain.

**2.3.1Design** Unlike client-server database management systems, the SQLite engine is not a standalone process with which the application program communicates. Instead, the SQLite [library](http://en.wikipedia.org/wiki/Library_%28computing%29) is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple [function calls](http://en.wikipedia.org/wiki/Subroutine), which reduces [latency](http://en.wikipedia.org/wiki/Latency_%28engineering%29) in database access as function calls within a single process are more efficient than [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication).

**2.3.2Features** SQLite implements most of the [SQL-92](http://en.wikipedia.org/wiki/SQL-92) standard for [SQL](http://en.wikipedia.org/wiki/SQL) but it lacks some features. A standalone program called sqlite3 is provided which can be used to create a database, define tables within it, insert and change rows, run queries and manage an SQLite database file. SQLite is a popular choice for local/client SQL storage within a [web browser](http://en.wikipedia.org/wiki/Web_browser) and within a [rich internet application framework](http://en.wikipedia.org/wiki/List_of_rich_internet_application_frameworks). This may be because SQLite's dynamically typed storage matches the [web browser](http://en.wikipedia.org/wiki/Web_browser)'s core languages of [Javascript](http://en.wikipedia.org/wiki/Javascript) and [XML](http://en.wikipedia.org/wiki/XML). SQLite uses an unusual [type system](http://en.wikipedia.org/wiki/Type_system) for an SQL-compatible DBMS. Instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is dynamically typed.

**SYSTEM ANALYSIS**

1. **SYSTEM ANALYSIS**

**3.1 SOFTWARE REQUIREMENTS**

* Java
* Android SDK 1.5 or later
* Eclipse Ganymede IDE
* Operating System can be Windows XP, LINUX and Mac etc.

**3.2 HARDWARE REQUIREMENTS**

1. PROCESSOR : (min) P2 Processor

2. RAM : (min) 128

3. HARD DISK : 100 MB

### 3.3 FEASIBILITY STUDY

An important outcome of the preliminary investigation is the determination that system requested is feasible. This is to identify the objectives of a new system. Before solving a problem one must know what the problem is. The study is carried out by a small group of people who are familiar with system analysis and design process. Fact finding techniques are used to gather the required information.

The three major areas consider while determining the feasibility of the project are

1. Economic Feasibility
2. Operational Feasibility
3. Technical Feasibility

**3.3.1 Economic Feasibility**

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**3.3.2 Operational Feasibility**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**3.3.3** **Technical Feasibility**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many-detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc.

A number of issues have to be considered while doing a technical analysis.

i) Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

ii) Find out whether the organization currently possesses the required technologies:

Is the required technology available with the organization?

If so is the capacity sufficient?

For instance -

“Will the current printer be able to handle the new reports and forms required for the new system?”

**SYSTEM DESIGN**

**4. SYSTEM DESIGN**

**4.1 ARCHITECTURE DIAGRAM**

Architecture diagram is a [diagram](http://en.wikipedia.org/wiki/Diagram) of a [system](http://en.wikipedia.org/wiki/System), in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

Fig 4.1 Architecture of Call History

Call History

User

Android SDK

SQLite

Java program

Get call logs

A Call History user for who the application looks like an user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and hence all its features apply here as well such as platform independence, data hiding, portable etc.

**4.2 DATA FLOW DIAGRAMS**

A **data-flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system). DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

`A DFD is also known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design.

So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**4.2.1 DFD Symbols**

In the DFD, there are four symbols

1. A Square defines a source or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows.
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store ,data at rest or a temporary repository of data

Process that transforms data flow

Source or destination of data

Data flow

Data store

**4.3 UNIFIED MODELING LANGUAGE (UML)**

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

#### Building Blocks of the UML:

#### 

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

**Things in the UML:**

There are four kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

**UML Diagram For GSearch:**

**USECASE DIAGRAMS:**

**Use case Diagram user:**



**Note: Delete the Todo List from the above use case diagram**

**Sequence Diagram:**



**CLASS DIAGRAM:**



**SYSTEM CODING & IMPLEMENTATION**

**5. SYSTEM CODING & IMPLEMENTATION**

**5.1 Coding:**

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.DialogInterface;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.KeyEvent;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.EditText;

**public** **class** Uid **extends** Activity **implements** OnClickListener {

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

View newButton = findViewById(R.id.*ss*);

newButton.setOnClickListener(**this**);

View aboutButton = findViewById(R.id.*clear\_button*);

aboutButton.setOnClickListener(**this**);

View exitButton = findViewById(R.id.*exit\_button*);

exitButton.setOnClickListener(**this**);

//View insertdataButton = findViewById(R.id.InsertData\_button);

// insertdataButton.setOnClickListener(this);

}

@Override

**public** **void** onClick(View v) {

EditText entryEditText = (EditText)findViewById(R.id.*entry*);

String data=entryEditText.getText().toString();

**switch** (v.getId()) {

**case** R.id.*ss*:

Intent k = **new** Intent(**this**, TabView.**class**);

k.putExtra("search",data);

k.putExtra("tab","0");

startActivity(k);

**break**;

**case** R.id.*clear\_button*:

entryEditText.setText("");

**break**;

**case** R.id.*exit\_button*:

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

builder.setMessage("Are you sure you want to exit?")

.setCancelable(**false**)

.setPositiveButton("Yes", **new** DialogInterface.OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** id) {

Uid.**this**.finish();

}

})

.setNegativeButton("No", **new** DialogInterface.OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

AlertDialog alert = builder.create();

alert.setTitle("Exit");

alert.setIcon(R.drawable.*icon*);

alert.show();

**break**;

}

}

**public** **boolean** onKeyDown(**int** keyCode, KeyEvent event) {

**if** (keyCode == KeyEvent.*KEYCODE\_BACK*)

{

Intent intent = **new** Intent(Intent.*ACTION\_MAIN*);

intent.addCategory(Intent.*CATEGORY\_HOME*);

intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

startActivity(intent);

**return** **true**;

}

**return** **super**.onKeyDown(keyCode, event);

}

import java.io.InputStream;

import java.util.ArrayList;

import android.app.Activity;

import android.content.ContentResolver;

import android.content.ContentUris;

import android.content.Intent;

import android.database.Cursor;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.net.Uri;

import android.os.Bundle;

import android.provider.ContactsContract;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.AdapterView.OnItemClickListener;

public class Contacts extends Activity {

ArrayList<String>al=new ArrayList<String>();

ArrayList<String>al2=new ArrayList<String>();

ArrayList<String>al3=new ArrayList<String>();

private static final Uri URI = ContactsContract.Contacts.CONTENT\_URI;

private static final String ID = ContactsContract.Contacts.\_ID;

private static final Uri PURI = ContactsContract.CommonDataKinds.Phone.CONTENT\_URI;

private static final String CID = ContactsContract.CommonDataKinds.Phone.CONTACT\_ID;

private static final String DNAME = ContactsContract.Contacts.DISPLAY\_NAME;

private static final String HPN = ContactsContract.Contacts.HAS\_PHONE\_NUMBER;

private static final String PNUM = ContactsContract.CommonDataKinds.Phone.NUMBER;

private String id,data=null;

View row;

private String name;

private int phcounter;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.contactreg);

Bundle extras=getIntent().getExtras();

data=extras.getString("search");

String s="%"+data+"%";

ListView lv=(ListView)findViewById(R.id.ListView01);

ContentResolver cr = getContentResolver();

Cursor cu = cr.query(URI, null, "DISPLAY\_NAME LIKE '"+s+"'", null, ContactsContract.Contacts.DISPLAY\_NAME + " ASC ");

if (cu.getCount() > 0) {

// Loop over all contacts

while (cu.moveToNext()) {

id = cu.getString(cu.getColumnIndex(ID));

al3.add(id);

name = cu.getString(cu.getColumnIndex(DNAME));

phcounter = 0;

if (Integer.parseInt(cu.getString(cu.getColumnIndex(HPN))) > 0) {

Cursor pCur = cr.query(PURI, null, CID + " = ?", new String[]{id}, null);

// String iddd=pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.\_ID));

// al3.add(iddd);

while (pCur.moveToNext()) {

String number = pCur.getString(pCur.getColumnIndex(PNUM));

al.add(name);

al2.add(number);

phcounter ++;

}

pCur.close();

}

}

}

lv.setAdapter(new ImageAdapter());

lv.setOnItemClickListener(new OnItemClickListener()

{

public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,

long arg3) {

Intent it=new Intent(Contacts.this,Call.class);

it.putExtra("phone",al2.get(arg2));

startActivity(it);

}

});

}

public class ImageAdapter extends BaseAdapter

{

@Override

public int getCount() {

// TODO Auto-generated method stub

return al.size();

}

@Override

public Object getItem(int position) {

// TODO Auto-generated method stub

return position;

}

@Override

public long getItemId(int position) {

// TODO Auto-generated method stub

return position;

}

@Override

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

//ContentResolver cr=getContentResolver();

LayoutInflater inflater=getLayoutInflater();

row=inflater.inflate(R.layout.contactsresult, parent,false);

ImageView iv=(ImageView)row.findViewById(R.id.contact\_image);

iv.setScaleType(ImageView.ScaleType.CENTER\_CROP);

iv.setPadding(8, 8, 8, 8);

TextView tv=(TextView)row.findViewById(R.id.contact\_name);

TextView tv2=(TextView)row.findViewById(R.id.phone\_number);

tv.setText(al.get(position));

tv2.setText(al2.get(position));

/\*Bitmap b=loadContactPhoto(cr,Long.parseLong(al3.get(position)));

if(b!=null)

{

iv.setImageBitmap(b);

}\*/

return row;

}

}

public static Bitmap loadContactPhoto(ContentResolver cr, long id) {

Uri uri = ContentUris.withAppendedId(ContactsContract.Contacts.CONTENT\_URI, id);

InputStream input = ContactsContract.Contacts.openContactPhotoInputStream(cr, uri);

if (input == null) {

return null;

}

return BitmapFactory.decodeStream(input);

}

}

**import** java.util.ArrayList;

**import** android.app.Activity;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.database.Cursor;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.os.Bundle;

**import** android.provider.MediaStore;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.AdapterView;

**import** android.widget.BaseAdapter;

**import** android.widget.GridView;

**import** android.widget.ImageView;

**import** android.widget.AdapterView.OnItemClickListener;

**public** **class** Camera **extends** Activity {

ArrayList<String> al=**new** ArrayList<String>();

**int** columnIndex=0;

String pavan=**null**;

Cursor c=**null**;

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*cameratab*);

Bundle extras=getIntent().getExtras();

pavan=extras.getString("search");

pavan="%"+pavan+"%";

c = managedQuery( MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*,

**null**, // Which columns to return

"TITLE LIKE '"+pavan+"'", // Return all rows

**null**,

MediaStore.Images.Media.*\_ID*);

**int** i=c.getColumnIndex(MediaStore.Images.Media.*DATA*);

columnIndex = c.getColumnIndexOrThrow(MediaStore.Images.Media.*\_ID*);

**if**(c.moveToFirst())

{**do**{

al.add(c.getString(i));

}**while**(c.moveToNext());

}

GridView gridview = (GridView) findViewById(R.id.*sdcard*);

gridview.setAdapter(**new** MyAdapter(**this**));

gridview.setOnItemClickListener(**new** OnItemClickListener()

{

@Override

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2,

**long** arg3) {

Intent it=**new** Intent(Camera.**this**,ImageAdapter2.**class**);

it.putStringArrayListExtra("a1", al);

it.putExtra("sel", **new** Integer(arg2).toString());

startActivity(it);

}

});

}

**public** **class** MyAdapter **extends** BaseAdapter {

**private** Context mContext;

**public** MyAdapter(Context c) {

// **TODO** Auto-generated constructor stub

mContext = c;

}

@Override

**public** **int** getCount() {

// **TODO** Auto-generated method stub

**return** al.size();

}

@Override

**public** Object getItem(**int** arg0) {

// **TODO** Auto-generated method stub

**return** al.get(arg0);

}

@Override

**public** **long** getItemId(**int** arg0) {

// **TODO** Auto-generated method stub

**return** arg0;

}

@Override

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

// **TODO** Auto-generated method stub

View grid;

**if**(convertView==**null**){

grid = **new** View(mContext);

LayoutInflater inflater=getLayoutInflater();

grid=inflater.inflate(R.layout.*mygrid*, parent, **false**);

}**else**{

grid = (View)convertView;

}

ImageView imageView = (ImageView)grid.findViewById(R.id.*imagepart*);

//TextView textView = (TextView)grid.findViewById(R.id.textpart);

Bitmap bm = ShrinkBitmap(al.get(position), 300, 300);

imageView.setImageBitmap(bm);

imageView.setScaleType(ImageView.ScaleType.*FIT\_CENTER*);

imageView.setPadding(8, 8, 8, 8);

**return** grid;

}

}

Bitmap ShrinkBitmap(String file, **int** width, **int** height){

BitmapFactory.Options bmpFactoryOptions = **new** BitmapFactory.Options();

bmpFactoryOptions.inJustDecodeBounds = **true**;

Bitmap bitmap = BitmapFactory.*decodeFile*(file, bmpFactoryOptions);

**int** heightRatio = (**int**)Math.*ceil*(bmpFactoryOptions.outHeight/(**float**)height);

**int** widthRatio = (**int**)Math.*ceil*(bmpFactoryOptions.outWidth/(**float**)width);

**if** (heightRatio > 1 || widthRatio > 1)

{

**if** (heightRatio > widthRatio)

{

bmpFactoryOptions.inSampleSize = heightRatio;

} **else** {

bmpFactoryOptions.inSampleSize = widthRatio;

}

}

bmpFactoryOptions.inJustDecodeBounds = **false**;

bitmap = BitmapFactory.*decodeFile*(file, bmpFactoryOptions);

**return** bitmap;

}

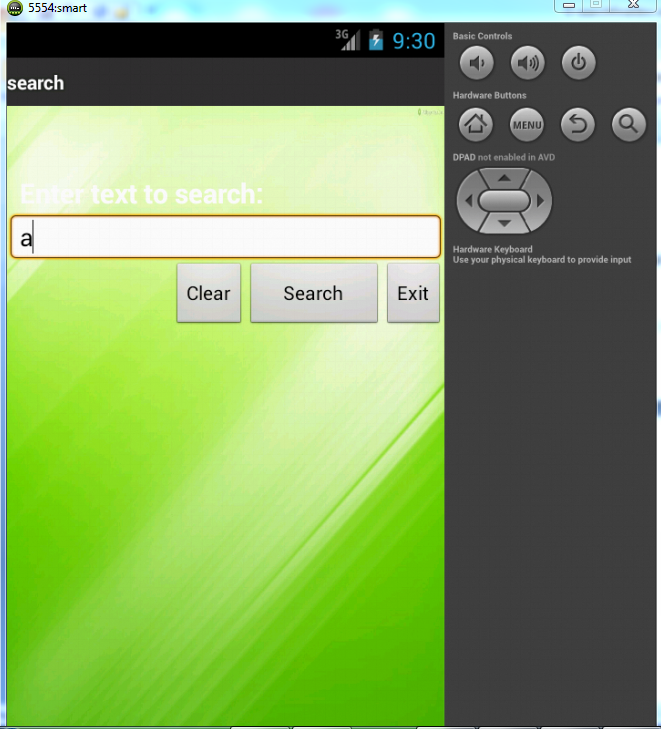
}

**5.2 Screen Shots:**

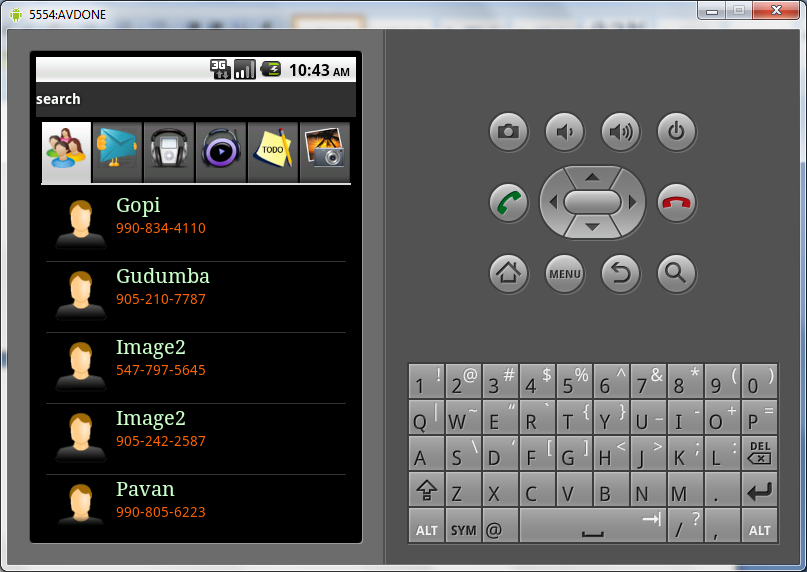
**Start Up Screen**

**Home Screen**

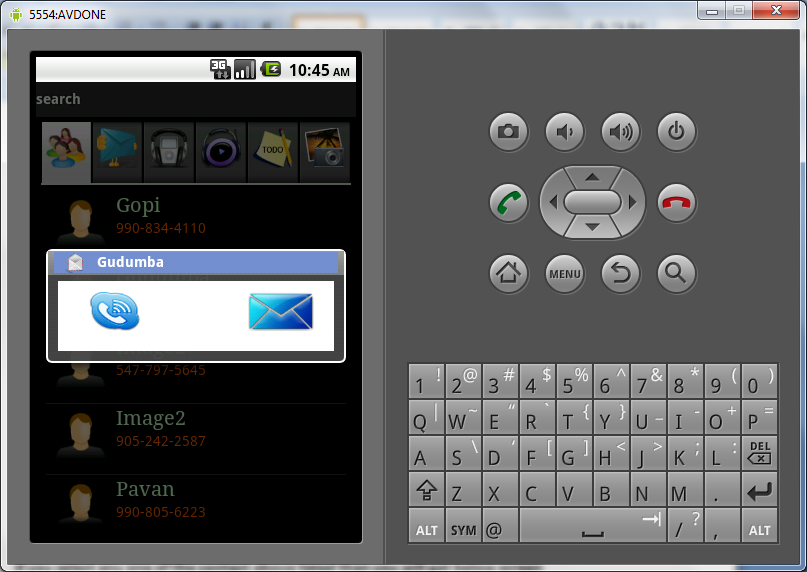
**If you enter any text in the given text field and if you press the Search button then it will search the content by the given letter.**

****

**Contacts Tab**

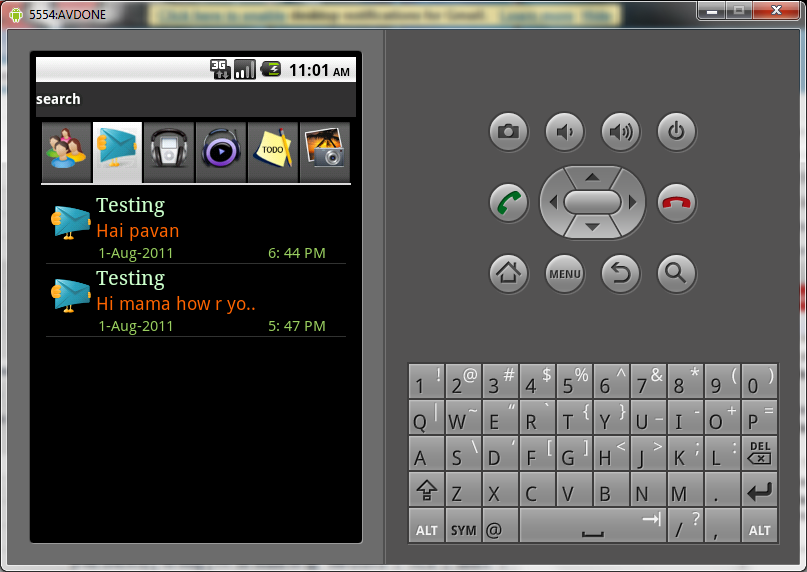
****

**If you select any one of the contact above listed than you will get below screen**

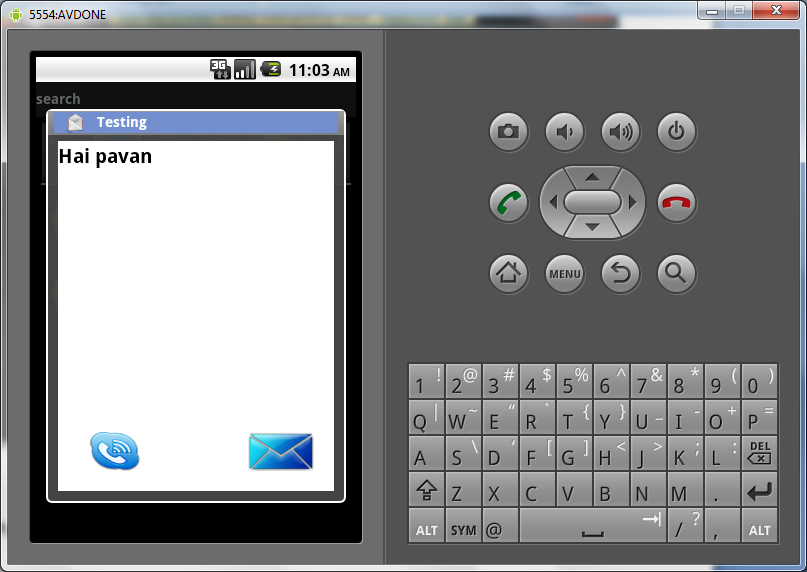
****

**This is for make a call or sending a message to particular no**

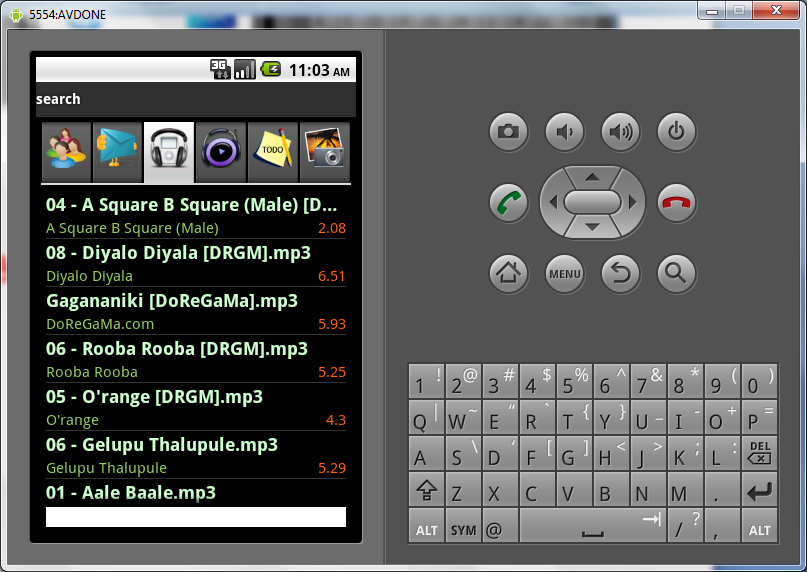
**Messages**

****

**If you click on the particular message below window will open**

****

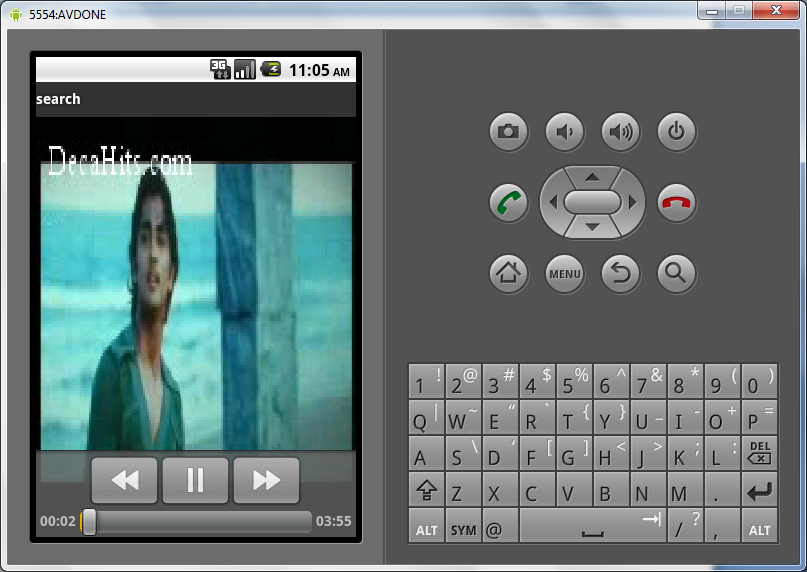
**Audio**

****

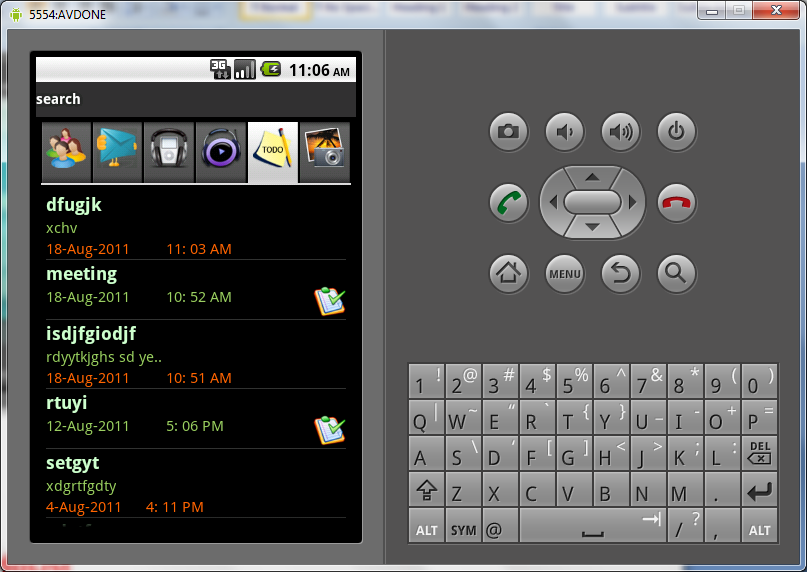
**Video**

****

**If you click on the video that will play in next window.**

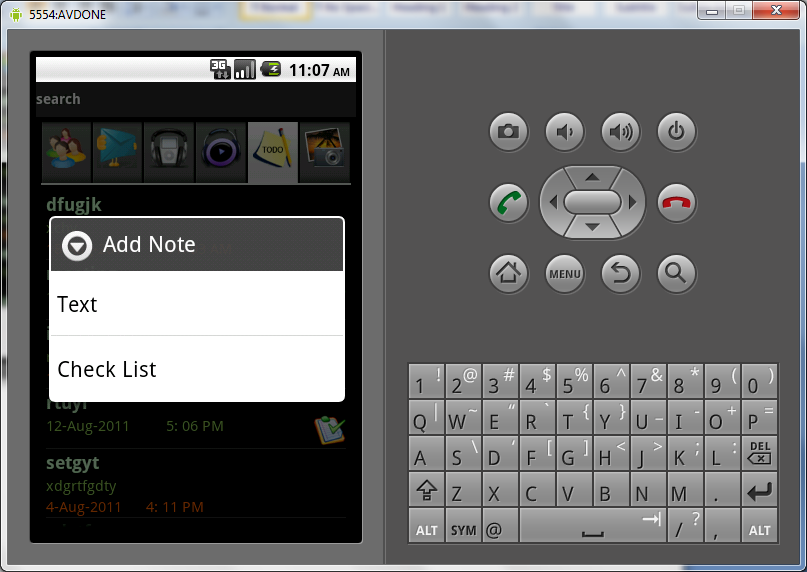
****

**TodoList**

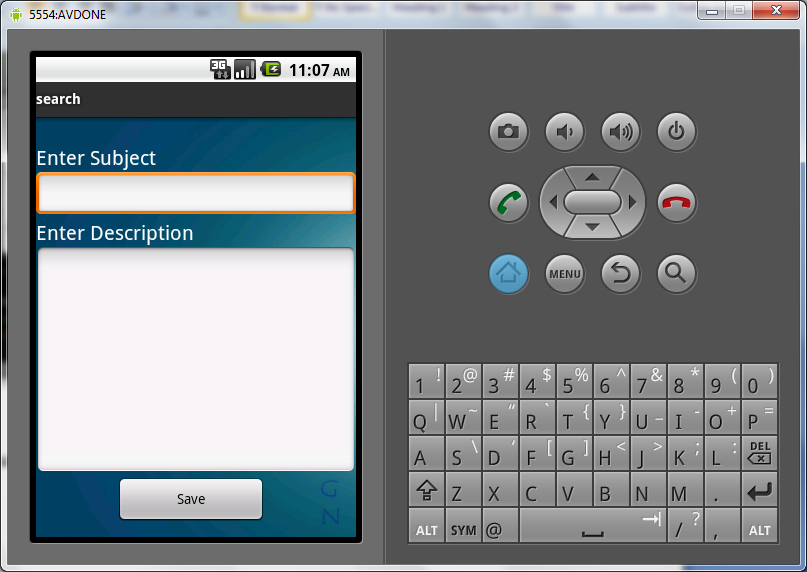
****

**If you want to enter the text or check list press the menu button than you will get**

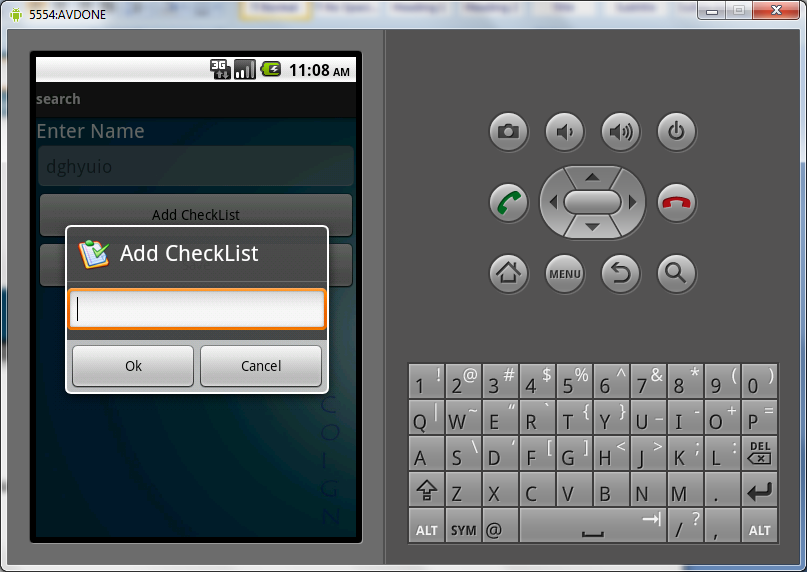
****

****

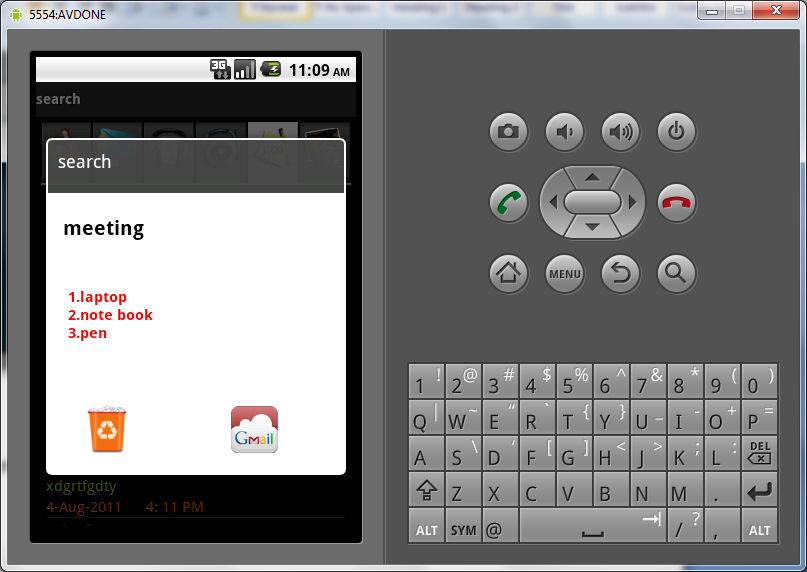
**If you click on the text**

****

**If you click on the check list**

****

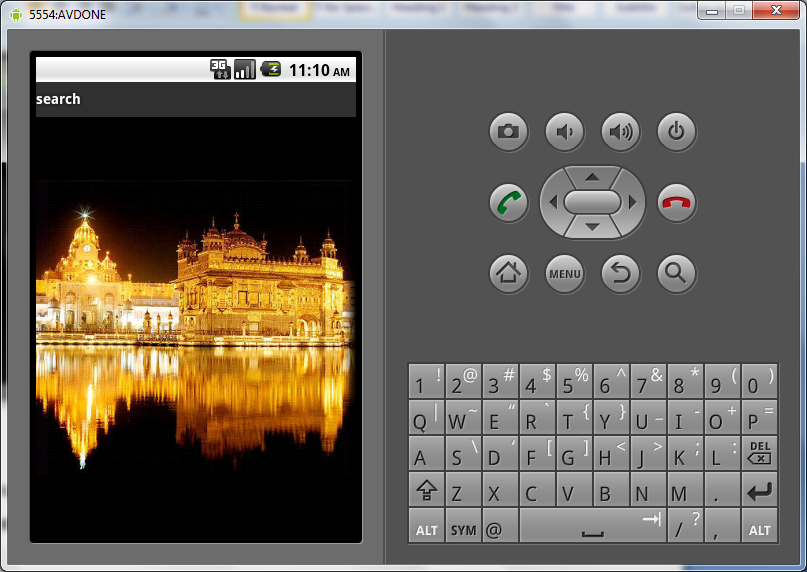
**After adding this check list if you click on the content**

****

Images



if you select the specific image in the grid then that image will appear



Here we have slide option also if we slide this than the next images will appear

**5.3 Method of Implementation**

Implementation literally means to put into effect or to carry out. The system implementation phase of the software deals with the translation of the design specifications into the source code. The ultimate goal of the implementation is to write the source code and the internal documentation so that it can be verified easily. The code and documentation should be written in a manner that eases debugging, testing and modification. System flowcharts, sample run on packages, sample output etc. Is part of the implementation?

An effort was made to satisfy the following goals in order specified.

• Minimization of Response Time.

• Clarity and Simplicity of the Code.

• Minimization of Hard-Coding.

Various types of bugs were discovered while debugging the modules. These ranged from logical errors to failure on account of various processing cases.

**TESTING AND VALIDATION**

**6. TESTING AND VALIDATION**

The development of software systems involves of a series of production activities where opportunities for injection of human fallibilities are enormous. Errors may begin to occur at the very inception of the process where the objectives may be erroneously or imperfectly specified, as well as in later design and development stages. Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity.

# TESTING TECHNIQUES

Testing is the process of executing a program with the intention of finding errors. The various test strategies used for testing the software are as follows.

**6.1 Unit Testing**

Unit testing focuses on verification effort on the smallest unit of the software design module. The main goal is to make sure that every source statement and logic path has been executed correctly at least once. The output of this stage is the source code.

**6.2 Integration Testing**

In Integration testing, we find errors that have occurred during the integration. After testing each module, which is then integrated into subsystems and then to form the entire system on which integration testing is performed. The goal of testing is to detect the design errors, while focusing on the testing the interconnection between modules.

## 6.3 Validation Testing

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

**6.4 System Testing**

In this testing, the system is tested for the errors after coupling all the modules together The system is tested against the specified requirements to see if all the requirements are met and the system performs as specified by the requirements.

|  |  |
| --- | --- |
| Test case 1: Database creation and updating. | **Priority (H, L):** High |
| Test Objective: For database creation. | |
| Test Description: User selects book name | |
| Requirements Verified: Yes | |
| Test Environment: Application must be deployed in android mobile phone or emulator. | |
| Test Setup/Pre-Conditions: | |
|  | |
| Actions | **Expected Results** |
| The user will select book with details | Stores the book details in database |
| Pass: Yes Conditions pass: Yes Fail: No | |
| Problems / Issues: NIL | |
| Notes: Successfully Executed | |

**6.5 Test cases**

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database or other common repository.

|  |  |
| --- | --- |
| Test case 2: Searching for books | **Priority (H, L):** High |
| Test Objective: For database creation. | |
| Test Description: User enters book name | |
| Requirements Verified: Yes | |
| Test Environment: Application must be deployed in android mobile phone or emulator. | |
| Test Setup/Pre-Conditions: | |
| Actions | **Expected Results** |
| The user enter book name | Searches books based on pattern matching. |
| Pass: Yes Conditions pass: Yes Fail: No | |
| Problems / Issues: NIL | |
| Notes: Successfully Executed | |

|  |  |
| --- | --- |
| Test case 2: Adding books into internal database | **Priority (H, L):** High |
| Test Objective: For database creation. | |
| Test Description: User enters book name | |
| Requirements Verified: Yes | |
| Test Environment: Application must be deployed in android mobile phone or emulator. | |
| Test Setup/Pre-Conditions: | |
| Actions | **Expected Results** |
| The user selects required book among the list of books | Add the book into database |
| Pass: Yes Conditions pass: Yes Fail: No | |
| Problems / Issues: NIL | |
| Notes: Successfully Executed | |

**CONCLUSION**

**7. CONCLUSION**

* 1. **CONCLUSION**

This application is used for getting call history. This application is build in Google mobiles using Android SDK. It is a tool developed for android platform, which is used to search various books and their related information within the mobile. This is an advantage when compared to existing system because a single mobile piece is enough for deploying the application .As this is a mobile application one can easily search for required information. One can search for books whenever one wants to without waiting for some system. This makes this application efficient, convenient and easy to use along with providing maximum user satisfaction which is the key aspect for any developer.

**7.2 FUTURE** **ENHANCEMENTS**

Any project that has been already developed can always be improved further

for better efficiency, better performance, easy understanding and important of all satisfy customer/user to a higher extent.

The future enhancements that can be done to this project are

* A dictionary can also be provided with this application for quick reference whenever needed while reading a book.
* A Google map can also be provided in order to know the exact location of the book store.
* E-book downloading can also be provided that can be an extension of this.

**REFERENCES**

## 8. REFERENCES

1. Herbert Schildt.2008 ,”Java Complete Reference”, Tata McGraw-Hill ,

7th Edition, pp.177-180 .

1. Grady Booch, James Rambaugh.1998,“UnifiedModeling Language User Guide”, Addison Wesley Publishing, chapter 8-31.

**WEBSITES:**  **REFERRED URLS:**

www.android.com http://developer.android.com/index.html

www.google.com http://en.wikipedia.org/wiki/SQLite