CHAPTER 1

Introduction

1. INTRODUCTION

1.1 Purpose And Motivation:

The main purpose of this vision document is to list the requirements of the college. This document also helps us to collect and analyze the ideas gathered for the project. This vision document will be subject to change, if more requirements are added to the project. This document is mainly prepared to set stage for the design phase of the project.

Android programming is a technology which is being used a lot in the now a days. My interest to learn this technology has prompted me to take up this project, which would set the stage for the applications, I would be developing in the future.

1.2 Project Overview:

The Android College App project is an implementation of a general College websites which helps the parents to get the availability of seats in college. This project also covers various features like Internal Assessments, Location of the .etc.

1.3 ABSTRACT:

Since Android Mobile devices have become more and more powerful and distributive, Mobiles are devices. With the invent of SGP Android Mobile App., Parents can get the numerous features and program offered. This App also provides Real time events and news updates. With the help of this App,We can get the information like, Location of the college and details of Admission process criteria and facilities and faculty (Academic staff). A mobile app is an Interactive Application.

1.4 EXISTING SYSTEM:

The Existing SGP website has many shortcomings associated with it. In this existing website parents and students can‘t get Internal Assessment Marks and other drawbacks like existing website is a static website, If parents wants their children marks and activities parents should come to Institution this causes waste of time, Using SGP Android Mobile App will gives Useful information their parents like., IA's Marks and also events going in college.

1.5 PROPOSED SYSTEM:

The SGP Android Mobile App maintains the database centrally giving to the student information required from college, whenever required. Through SGP Android mobile App parents or students is able to get required information from his home/office anywhere in the world it doesn't require parents or students should to go to the college thus saving the time.

1.6 Objectives

1.To provide the information about the currently available seats in college.

2. Easy and secure to get Internal Assessments marks.

3. Any one can able to get the Location of the college.

4. To provide faculty details according to Departments.

CHAPTER 2

LITERATURE SURVEY

2.1 The Android

2.1. 1 Introduction to Android :

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

2.1.2 Features of Android :

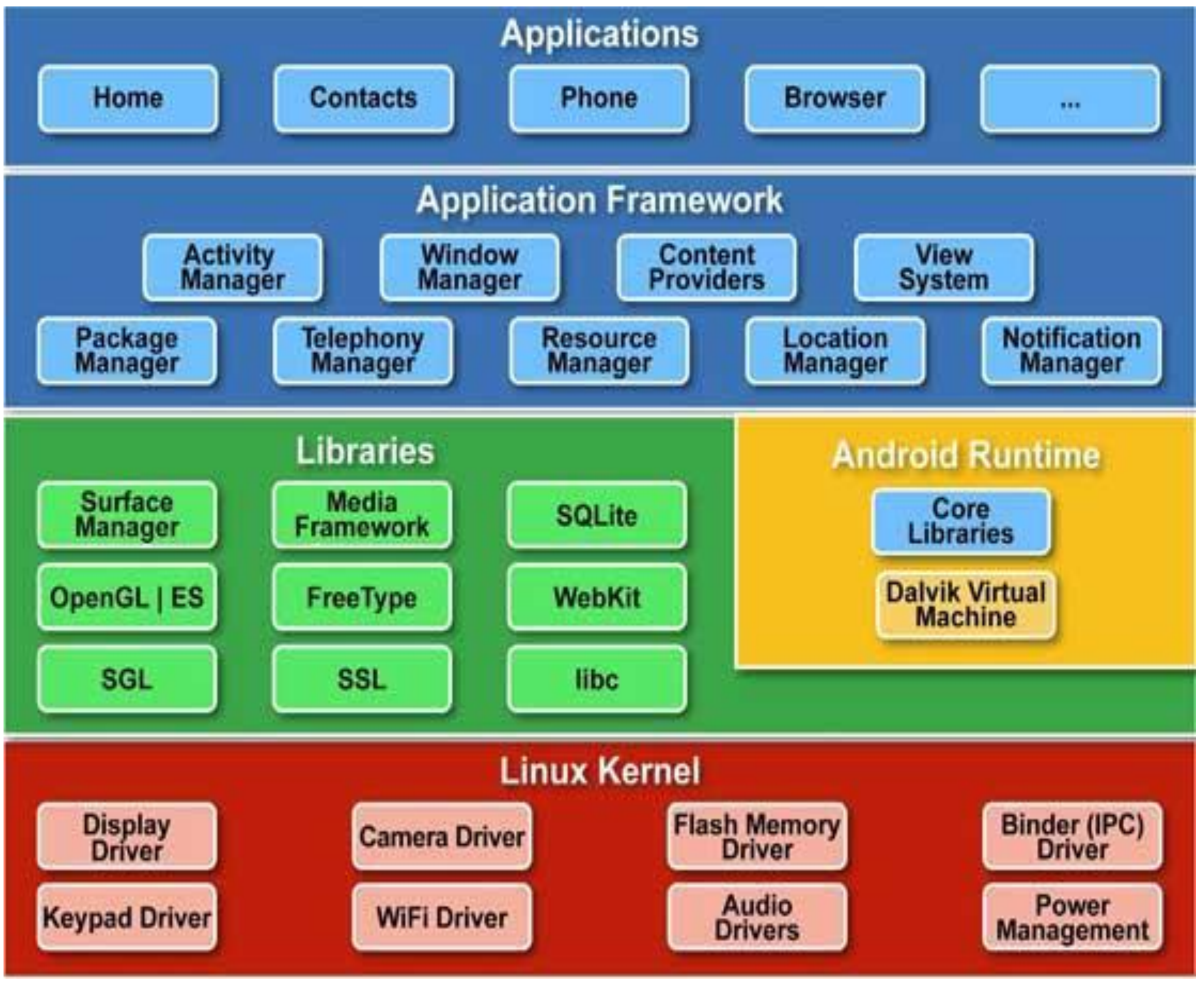
Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below:

- ***Beautiful UI :****Android OS basic screen provides a beautiful and intuitive user interface.*
- ***Connectivity :***GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
- ***Storage :*** *SQLite, a lightweight relational database, is used for data storage purposes.*
- ***Media support :*** *H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF, and BMP.*
- ***Messaging :****SMS and MMS.*
- ***Web browser :****Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3***.**
- ***Multi-touch :****Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.*
- ***Multi-tasking :****User can jump from one task to another and same time various application can run simultaneously***.**

- ***Resizable widgets*** *:Widgets are resizable, so users can expand them to show more content or shrink them to save space.*
- ***Multi-Language*** *:Supports single direction and bi-directional text.*
- ***GCM*** *:Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.*
- ***Wi-Fi Direct*** *:A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.*
- ***Android Beam*** *:A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.*

2.1.3Android Architecture :

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



- ***Linux kernel***:
  At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.
- ***Libraries*** **:**
  On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.
- ***Android Runtime :***
  This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called ***Dalvik Virtual Machine*** *which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.*
- ***Application Framework :***
  The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.
- ***Applications :***
  You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

2.1.4 Android application components :

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

- ***Activities :****They they dictate the UI and handle the user interaction to the smartphone screen.*
- ***Services :*** *They handle background processing associated with an application.*
- ***Broadcast Receivers :*** *They handle communication between Android OS and applications.*

- ***Content Providers*** ***:*** *They handle data and database management issues.*

➢ *Activities :*

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

- *An activity is implemented as a subclass of* ***Activity*** *class as follows:*

```
public class MainActivity extends Activity {
}
```

➢ *Services :*

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

- *A service is implemented as a subclass of* ***Service*** *class as follows:*

```
public class MyService extends Service {
}
```

➢ *Broadcast Receivers :*

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

- *A broadcast receiver is implemented as a subclass of* ***BroadcastReceiver****class and each message is broadcasted as an* ***Intent*** *object.*

```
public class MyReceiver extends BroadcastReceiver {
}
```

➢ *Content Providers :*

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolverclass. The data may be stored in the file system, the database or somewhere else entirely.

- *A content provider is implemented as a subclass of* ***ContentProvider****class and must implement a standard set of APIs that enable other applications to perform transactions.*

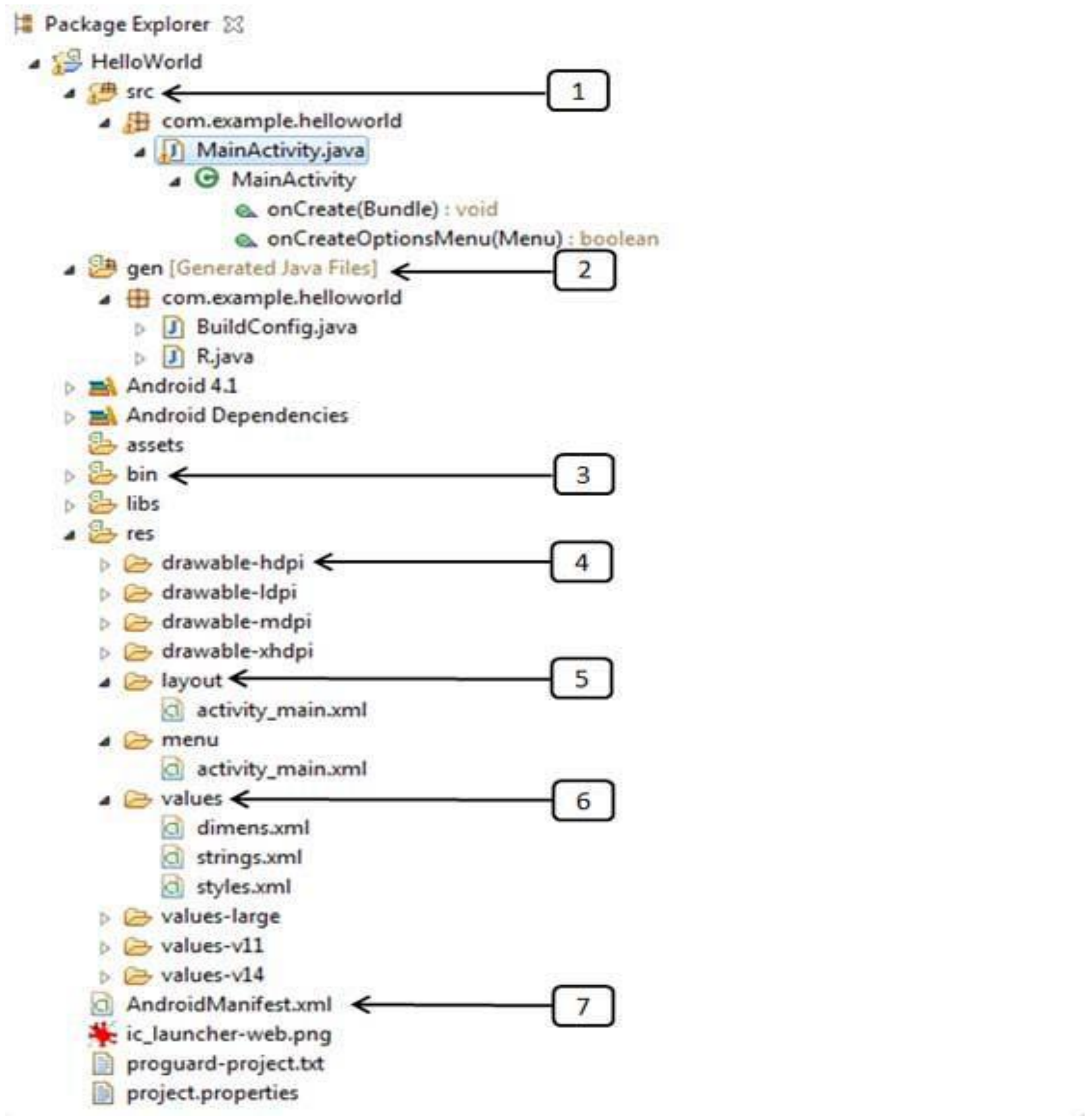
```
public class MyContentProvider extends ContentProvider {
 }
```

➢ *Additional Components :*

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are:

- ***Fragments :Represents a behavior or a portion of user interface in an Activity.***
- ***Views : UI elements that are drawn onscreen including buttons, lists forms etc.***
- ***Layouts: View hierarchies that control screen format and appearance of the views.***
- ***Intents : Messages wiring components together.***
- ***Resources: External elements, such as strings, constants and drawables pictures.***
- ***Manifest: Configuration file for the application.***

2.1.5 Anatomy of Android application :



```
Package Explorer
  HelloWorld
    src                                   1
      com.example.helloworld
        MainActivity.java
          MainActivity
            onCreate(Bundle) : void
            onCreateOptionsMenu(Menu) : boolean
    gen [Generated Java Files]            2
      com.example.helloworld
        BuildConfig.java
        R.java
    Android 4.1
    Android Dependencies
    assets
    bin                                   3
    libs
    res
      drawable-hdpi                       4
      drawable-ldpi
      drawable-mdpi
      drawable-xhdpi
      layout                              5
        activity_main.xml
      menu
        activity_main.xml
      values                              6
        dimens.xml
        strings.xml
        styles.xml
      values-large
      values-v11
      values-v14
    AndroidManifest.xml                   7
    ic_launcher-web.png
    proguard-project.txt
    project.properties
```

I. ***src*** *:This contains the .java source files for your project. By default, it includes anMainActivity.javasource file having an activity class that runs when your app is launched using the app icon.*

II. ***gen*** *:This contains the .R file, a compiler-generated file that references all the resources found in your project. You should not modify this file.*

III. ***bin*** *:This folder contains the Android package files .apkbuilt by the ADT during the build process and everything else needed to run an Android application.*

IV. ***res/drawable-hdpi*** *:This is a directory for drawable objects that are designed for high-density screens.*

V. ***res/layout* :***This is a directory for files that define your app's user interface.*

VI. ***res/values* :***This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.*

VII. ***AndroidManifest.xml* :** *This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.*

2.2 Introduction to java :

It is a product of Sun Microsystems. Java is C++ for the rest. It uses C++ style syntax but has been designed to be much easier to use. The core language is quite small and is extended by myriad packages of useful routines. In fact most of the fine programming in Java involves the right routine in the right package to do the job.

2.2.1 Other possible advantages of Java include :

1. *Object-oriented.*
2. *Platform independence.*
3. *Excellent networking capabilities.*
4. *No pointers, so it is easier to program and debug than C++.*
5. *Lets of support available in books and on the web.*
6. *Wide variety of Application Programmer Interfaces (APIs).*
7. *Implementations are available free of charge for many popular platforms.*

2.2.2 The java Platform :

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it’s a software only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

2.2.3 The Java platform has two components :

1. The Java Virtual (Java VM)
2 The Hava Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of readymade software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

The Java API is grouped into libraries (packages) of related components. The following figure depicts a Java program, such as an application or applet, that’s running on the Java *platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.*

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening probability.

2.2.4 Java Virtual Machine (JVM) :

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Java programming uses to produce byte codes and execute them. The first box indicates that the Java source code is located in a Java file that is processed with a Java compiler called javac. The java compiler produces a file called a class file, which contains the byte code. The class file is then loaded across the network or loaded locally on your machine into the execution environment in the Java virtual machine, which interprets and executes the byte code.

2.2.5 Java Architecture :

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

2.2.6 Compilation of code :

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code *is returned and compiled for one machine and interpreted on all machine. This is called Java Virtual Machine.*

2.3 Servelets:

2.3.1. Introduction :

Servlets provide a Java based solution used to address the problems concurrently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs and incomplete interfaces.

Servlets are objects that confirm to a specific interface that can be plugged into a Java based server. Servlets are to the server-side what applets are to the client-side –object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects. They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

2.3.2 Servlet :

Java servlet technology lets you define HTTP-specific servlet classes. A servlet class extends the capabilities of servers that host applications that are accessed by way of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

2.3.3 Servlet Lifecycle :

Each servlet has the same life cycle:

1. *A server loads and initializes the servlet.*
2. *The servlet handles zero or more client requests,*
3. *The server removes the servlet.*

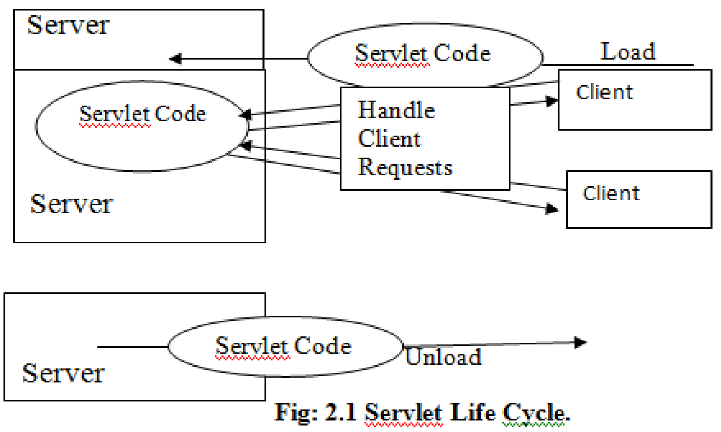


Fig: 2.1 Servlet Life Cycle.

2.4 JSP :

A Java Server Pages is a template for a Web page that uses Java code to generate HTML documentation dynamically. JSP are run in server side component known as JSP Container, which translates them into equivalent Java Servlets. For this reason Servlet and JSP pages are intimately related.

2.4.1 JSP Scripting Elements :

In order to be able to embed scripts of java code into HTML, JSP introduced new tags; three of them (the first, second and fourth) are called "Scripting elements"

1. *Expression tags <%= expression %>*
2. *Scriptlet tags <% scriptlet %>*
3. *Directive tags <%@ directive attribute="value" %>*
4. *Declarative tags <%! declarations %>*

1.Expression tags <%= expression %>

In our example, we used the tag <%= expression %> which is basically used to get the string value of this expression and write it to the body of the response. (Initially we will assume that it gets the string value of this expression and display it).

2. Scriptlet Tags <% scriptlet %>

The scriptlet tag <% scriptlet %> is the most commonly used tag, as it allows placing in an HTML code, a normal script of java code that you are used to write while implementing a desktop application.

In a scriptlet tag, you may declare variables, use loops, conditional operators and you can use the method out.println () to write a string value to the response body (or as assumed before; to display a string value on your result page). This out.println method functions almost exactly like the <%= expression %> tag.

3. Directive tags

<%@directive attribute = "value" %> is the syntax of the so called directives. As you can see they are somehow different, as they have attributes and attributes' values (they are not pure java scripts).A directive can be:

1. ***page**, which is used in importing external classes and providing information about the page.*
2. ***include** that is used to include the code written in another file. Note that it is not a must for this file's content to be a complete code representing a web page. This tag's function is like copying the content of a file (not necessarily a JSP), and pasting if in another file.*

4. Declarative tags

Declarative tags <%! declarations %> are used to define variables and methods. Note that you can define a variable in a scriptlet tag, but you can not define a method except in a declarative tag. Try to visualize the difference by assuming that the code written in all scriptlet tags of the whole JSP is placed in a main method, while code written in declarative tags is placed outside the main method (in the form of variables and methods).

Forwarding Requests:

With the <jsp:forward>tag, you can redirect the request to any JSP, servlet, or static HTML page within the same context as the invoking page. This effectively halts processing of the current page at the point where the redirection occurs, although all processing up to that point still takes place:
<jsp:forward page= "somePage.jsp"/>

2.5 JDBC

Anyone can write a single program using the JDBC API, and the JDBC is a Java Api for executing SQL, statements(As a point of interest JDBC is trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java Programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually program will be able to send SQL .statements to the appropriate database. The Combination of Java and JDBC lets a programmer writes it once and run it anywhere.

What Does JDBC Do?

1. *Establish a connection with a database*
2. *Send SQL statements*
3. *Process the results*
4. *JDBC Driver Types*
5. *The JDBC drivers that we are aware of this time fit into one of four categories*
6. *JDBC-ODBC Bridge plus ODBC driver*
7. *Native-API party-java driver*
8. *JDBC-Net pure java driver*
9. *Native-protocol pure Java driver*

An individual database system is accessed via a specific JDBC driver that implements the java.sql.Driver interface. Drivers exist for nearly all-popular RDBMS systems, through few are available for free. Sun bundles a free JDBC-ODBC bridge driver with the JDK to allow access to a standard ODBC, data sources, such as a Microsoft Access database, Sun advises against using the bridge driver for anything other than development and very limited development.

JDBC drivers are available for most database platforms, from a number of vendors and in a number of different flavours. There are four driver categories

Type 01-JDBC-ODBC Bridge Drive

Type 01 drivers use a bridge technology to connect a java client to an ODBC database service. Sun's JDBC-ODBC bridge is the most common type 01 driver. These drivers implemented using native code.

Type 02-Native-API party-java Driver

*Type 02 drivers wrap a thin layer of java around database-specific native code libraries for Oracle databases, the native code libraries might be based on the OCI(Oracle call Interface) libraries, which were originally designed for **c/c++** programmers, Because type-02 drivers are implemented using native code. in some cases they have better performance than their all-java counter parts. They add an element of risk, however, because a defect in a driver's native code section can crash the entire server*

Type 03-Net-Protocol All-Java Driver

Type 03 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware component might use any type of driver to provide the actual database access. These drivers are all java, which makes them useful for applet deployment and safe for servlet deployment

Type-04-native-protocol All-java Driver

Type o4 drivers are the most direct of the lot. Written entirely in java, Type 04 drivers understand database-specific networking. protocols and can access the database directly without any additional software.

JDBC-ODBC Bridge

If possible use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, library, the ODBC driver library, and the database client library).

What Is The JDBC-ODBC Bridge?

The JDBC-ODBC Bridge is a Jdbc driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC.

2.6 The Process Flow

This diagram basically shows the process flow in our application.

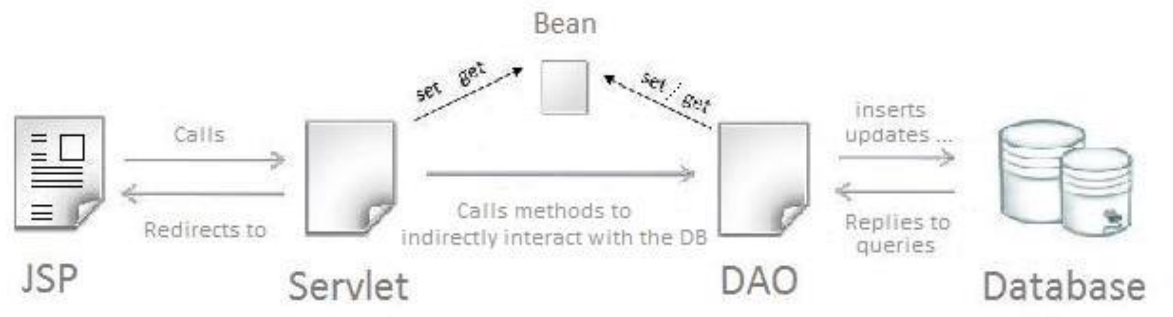


Fig: 2.2 Application Process Communicating with Database

CHAPTER 3

HARDWAREANDSOFTWARE REQUIREMENT SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

1.	CPU	: PENTIUM DUAL CORE processer
2.	Memory	: 2 GB
3.	Cache	: 2 MB
4.	Floppy Disk	: 1.44MB
5.	Hard Disk	: 80 GB
6.	Speed	: 2.2GHZ
7.	Display	: 15” Monitor(SVGA)
8.	Key Board	: Standard 108 Keys Enhanced Keyboard
9.	Mouse	: MS Serial Mouse

3.2 SOFTWARE REQUIREMENTS

1. Operating System	: Windows XP
2. IDE	: Eclipse Indigo
3. SDK	: Android SDK
4. JDK	:JAVA

CHAPTER 4

SYSTEM ANALYSIS

4.1 Study of the system

To provide flexibility to the users, the interfaces have been developed that are accessible through a Android App.

4.2 Number of Modules :

After careful analysis the system has been identified to have the following modules:

- ❖ *Campus Map Module*
- ❖ *Event Module*
- ❖ *IA Marks module*
- ❖ *Faculty module*
- ❖ *Departments*
- ❖ *About us module*

- ➢ *Campus Map Module :This will give you information like., where our locates and it uses a GPS to location this means user can get the route address of the college how far is there from user actual place.*

- ➢ *Event Module :In this 3 sub modules are there given below.*

I. *Past Events*
II. *Current Events*
III. *Future Events*

- ✓ *Past Events : In this sub module user can get events details, which events have been done previously in college.*
- ✓ *Current Events : In this sub module user can see the currently going on events in our college.*
- ✓ *Future Events : This sub module will give you what will be future events or upcoming will be in the college.*

➢ *IA Marks Module : This module will give you Internal Assessment Marks according to the 1st , 2nd ,3rd internals.*

Main advantage of this module is parents also can view their children IA marks.

➢ *Faculty Module : This module will give you faculty details according to department. The faculty details contains the faculty name and designation.*

➢ *Department Module : This module display the list of departments of our college.*

CHAPTER 5

DIAGRAMS

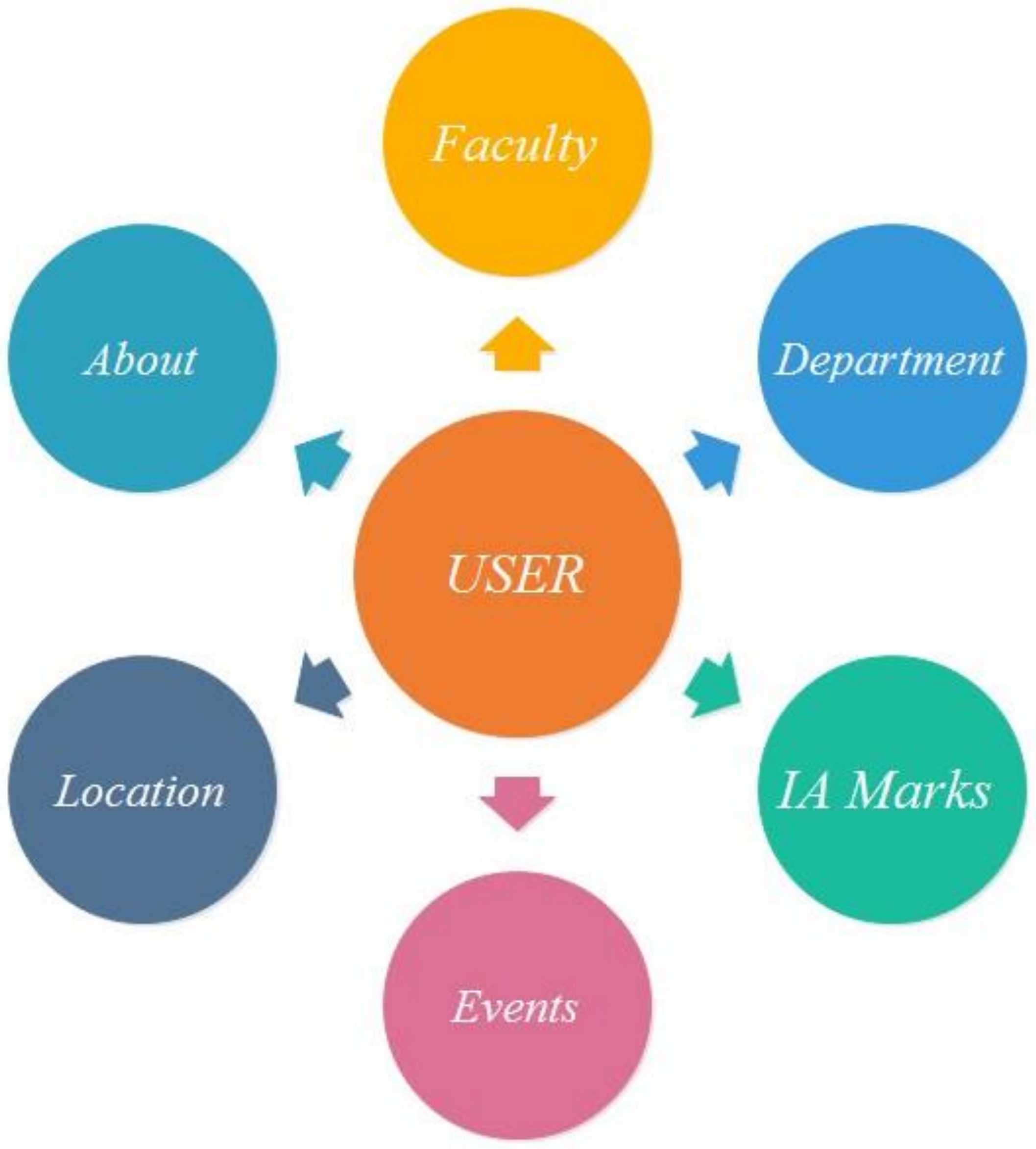


Figure 5.1 : Circular Modular Diagram

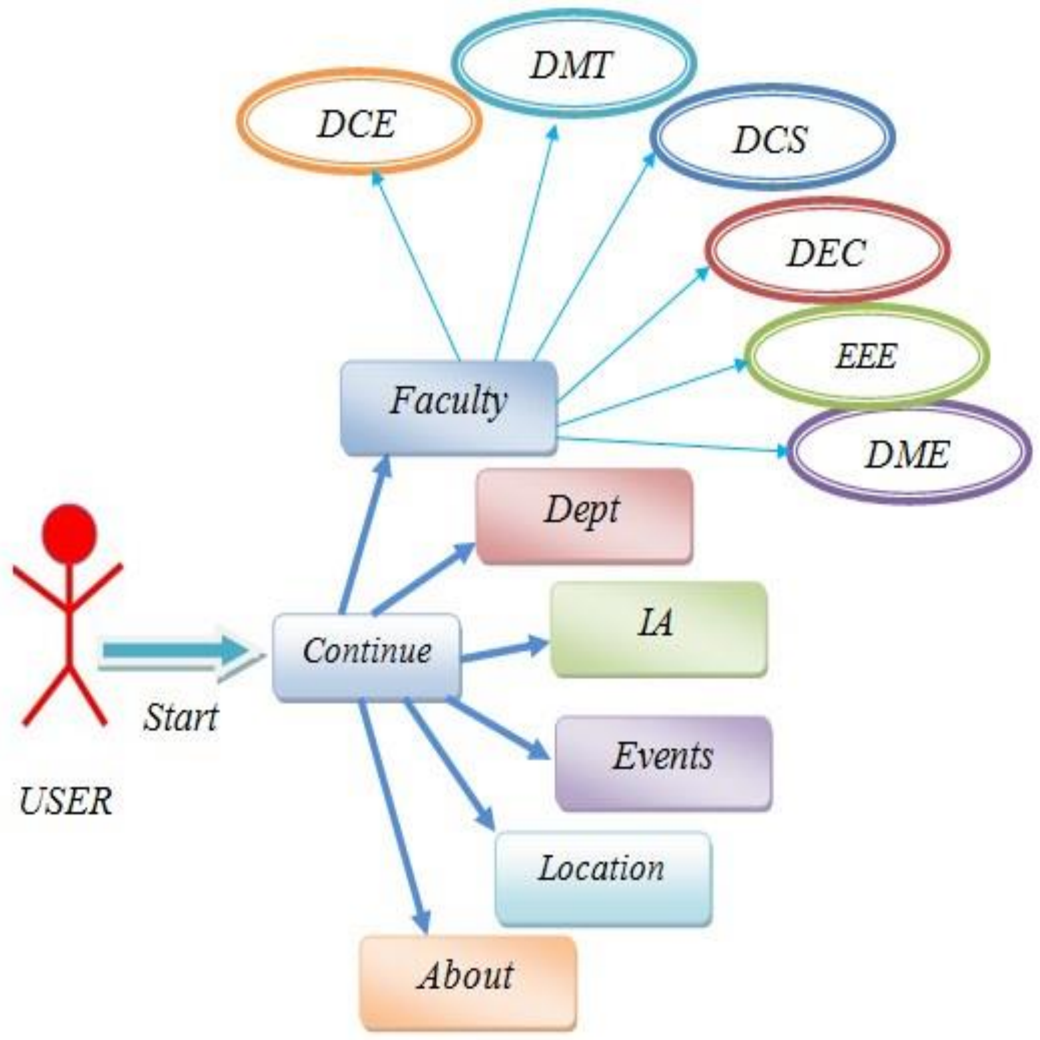


Figure 5.2 ER Diagram

CHAPTER 6

PROGRAM CODE

//AboutActivity

```
package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class AboutActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_about);}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.about, menu);
                return true;}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);}
}
```

//CivilActivity

```
package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class CivilActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_civil);
                WebView wv = (WebView)findViewById(R.id.my_webview);
    wv.getSettings().setDomStorageEnabled(true);
    wv.getSettings().setBuiltInZoomControls(true);
    wv.getSettings().setSupportZoom(true);
    wv.getSettings().setSupportMultipleWindows(true);
    wv.getSettings().setLoadWithOverviewMode(false);
    wv.getSettings().setJavaScriptEnabled(true);
    wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
    wv.getSettings().setUseWideViewPort(false);
     wv.setWebViewClient(new WebViewClient() {
    @Override
     public boolean shouldOverrideUrlLoading(WebView view, String url) {
          view.loadUrl(url);
          return true;
       } });
   ip ip = new ip();
```

```
        String ip_address=ip.getIp();
        wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/civil.jsp");}
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.civil, menu);
                return true;}
        @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
            }
}
```

//CseActivity

```
package com.sgp;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.webkit.WebView;

import android.webkit.WebViewClient;


public class CseActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

                    super.onCreate(savedInstanceState);

                    setContentView(R.layout.activity_cse);

                    WebView wv = (WebView)findViewById(R.id.my_webview);

      wv.getSettings().setDomStorageEnabled(true);

      wv.getSettings().setBuiltInZoomControls(true);

      wv.getSettings().setSupportZoom(true);

      wv.getSettings().setSupportMultipleWindows(true);

      wv.getSettings().setLoadWithOverviewMode(false);

      wv.getSettings().setJavaScriptEnabled(true);

      wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);

wv.getSettings().setUseWideViewPort(false);

 wv.setWebViewClient(new WebViewClient() {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

             view.loadUrl(url);

             return true;

          }
```

```
        });
        ip ip = new ip();
        String ip_address=ip.getIp();
    wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/cse.jsp");
        }

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

@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
        }
}
```

//Department

```
package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class Department extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_department);
            WebView wv = (WebView)findViewById(R.id.my_webview);
    wv.getSettings().setDomStorageEnabled(true);
    wv.getSettings().setBuiltInZoomControls(true);
    wv.getSettings().setSupportZoom(true);
    wv.getSettings().setSupportMultipleWindows(true);
    wv.getSettings().setLoadWithOverviewMode(false);
    wv.getSettings().setJavaScriptEnabled(true);
    wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
wv.getSettings().setUseWideViewPort(false);
wv.setWebViewClient(new WebViewClient() {
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
      }
```

```
        });
  ip ip = new ip();
        String ip_address=ip.getIp();
        wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/department.jsp");
        }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.department, menu);
                return true;
         }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
         }
}
```

//EceActivity

```
 package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class EceActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_ece);
                WebView wv = (WebView)findViewById(R.id.my_webview);
     wv.getSettings().setDomStorageEnabled(true);
     wv.getSettings().setBuiltInZoomControls(true);
     wv.getSettings().setSupportZoom(true);
     wv.getSettings().setSupportMultipleWindows(true);
     wv.getSettings().setLoadWithOverviewMode(false);
     wv.getSettings().setJavaScriptEnabled(true);
     wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
     wv.getSettings().setUseWideViewPort(false);
     wv.setWebViewClient(new WebViewClient() {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
          view.loadUrl(url);
          return true;
        }
        }); ip ip = new ip();
        String ip_address=ip.getIp();

        wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/ece.jsp");
           }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                    // Inflate the menu; this adds items to the action bar if it is present.
                    getMenuInflater().inflate(R.menu.ece, menu);
                    return true;
           }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                    // Handle action bar item clicks here. The action bar will
                    // automatically handle clicks on the Home/Up button, so long
                    // as you specify a parent activity in AndroidManifest.xml.
                    int id = item.getItemId();
                    if (id == R.id.action_settings) {
                              return true;
                    }
                    return super.onOptionsItemSelected(item);
           }
}
```

//EeeActivity

```
package com.sgp;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.webkit.WebView;

import android.webkit.WebViewClient;

public class EeeActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

                setContentView(R.layout.activity_eee);

                WebView wv = (WebView)findViewById(R.id.my_webview);

    wv.getSettings().setDomStorageEnabled(true);

    wv.getSettings().setBuiltInZoomControls(true);

    wv.getSettings().setSupportZoom(true);

    wv.getSettings().setSupportMultipleWindows(true);

    wv.getSettings().setLoadWithOverviewMode(false);

    wv.getSettings().setJavaScriptEnabled(true);

    wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);

wv.getSettings().setUseWideViewPort(false);

wv.setWebViewClient(new WebViewClient() {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

           view.loadUrl(url);

           return true;

      }

    });
```

```
        ip ip = new ip();
        String ip_address=ip.getIp();
 wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/eee.jsp");
            }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.eee, menu);
                return true;
            }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
            }
}
```

//EventActivity

```
 package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class EventActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.activity_event);
                    WebView wv = (WebView)findViewById(R.id.my_webview);
       wv.getSettings().setDomStorageEnabled(true);
       wv.getSettings().setBuiltInZoomControls(true);
       wv.getSettings().setSupportZoom(true);
       wv.getSettings().setSupportMultipleWindows(true);
       wv.getSettings().setLoadWithOverviewMode(false);
       wv.getSettings().setJavaScriptEnabled(true);
       wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
 wv.getSettings().setUseWideViewPort(false);
wv.setWebViewClient(new WebViewClient() {
          @Override
          public boolean shouldOverrideUrlLoading(WebView view, String url) {
             view.loadUrl(url);
             return true;
          }
       });
```

```
        ip ip = new ip();
        String ip_address=ip.getIp();
wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/events.jsp");
                        }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                        // Inflate the menu; this adds items to the action bar if it is present.
                        getMenuInflater().inflate(R.menu.event, menu);
                        return true;
            }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                        // Handle action bar item clicks here. The action bar will
                        // automatically handle clicks on the Home/Up button, so long
                        // as you specify a parent activity in AndroidManifest.xml.
                        int id = item.getItemId();
                        if (id == R.id.action_settings) {
                                    return true;
                        }
                        return super.onOptionsItemSelected(item);
            }
}
```

//FacultyActivity

```
 package com.sgp;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
public class FacultyActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_faculty);
            Button button=(Button)findViewById(R.id.button1);
          button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                  // TODO Auto-generated method stub
                  Intent i = new Intent(getApplicationContext(),CseActivity.class);
                  startActivity(i);
                }
              });
          Button button1=(Button)findViewById(R.id.button2);
          button1.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                  // TODO Auto-generated method stub
                  Intent i = new Intent(getApplicationContext(),EceActivity.class);
```

```
                    startActivity(i);
                }
            });
        Button button2=(Button)findViewById(R.id.button3);
        button2.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    Intent i = new Intent(getApplicationContext(),EeeActivity.class);
                    startActivity(i);
                }
            });
    Button button3=(Button)findViewById(R.id.button4);
        button3.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    Intent i = new Intent(getApplicationContext(),MechActivity.class);
                    startActivity(i);
                }
            });

        Button button4=(Button)findViewById(R.id.button5);
        button4.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    Intent i = new Intent(getApplicationContext(),MetActivity.class);
                    startActivity(i);
                    }
                });
            Button button5=(Button)findViewById(R.id.button6);
            button5.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),CivilActivity.class);
                        startActivity(i);
                    }
                });
        }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.faculty, menu);
                return true;
        }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
        }}
```

//IaActivity

```
 package com.sgp;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.NumberFormat;
import java.util.ArrayList;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
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.JSONException;
import org.json.JSONObject;
import android.app.Activity;
import android.os.Bundle;
import android.os.StrictMode;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class IaActivity extends Activity {
```

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

@Override

protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

                setContentView(R.layout.activity_ia);

StrictMode.setThreadPolicy(policy);

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

        Button button1 = (Button) findViewById(R.id.button2);

              button1.setOnClickListener(new View.OnClickListener()

              {

              public void onClick(View view)

              {

        String result = null;

                InputStream is = null;

                EditText editText = (EditText)findViewById(R.id.e1);

                String v1 = editText.getText().toString();

                EditText editText1 = (EditText)findViewById(R.id.e2);

                EditText editText2 = (EditText)findViewById(R.id.e3);

                ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();

nameValuePairs.add(new BasicNameValuePair("f1",v1));

                        try

                     {

                      HttpClient httpclient = new DefaultHttpClient();

                      HttpPost httppost = new
HttpPost("http://192.168.1.102/FLOWERS/sudha/select.php");

                      httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

                      HttpResponse response = httpclient.execute(httppost);

                      HttpEntity entity = response.getEntity();
```

```
                        is = entity.getContent();
Log.e("log_tag", "connection success ");
                           }
                           catch(Exception e)
                        {
                        Log.e("log_tag", "Error in http connection "+e.toString());
                        Toast.makeText(getApplicationContext(), "Connection fail",
Toast.LENGTH_SHORT).show();
 }
//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();
catch(Exception e)
                           {
                       Log.e("log_tag", "Error converting result "+e.toString());
                     Toast.makeText(getApplicationContext(), " Input reading fail",
Toast.LENGTH_SHORT).show();
}//parse json data
                  try{
                  JSONObject object = new JSONObject(result);
                             String ch=object.getString("re");
```

```
                              if(ch.equals("success"))
                              {
                             JSONObject no = object.getJSONObject("0");
                             String w= no.getString("f2");
                         long e=no.getLong("f3");
                        editText1.setText(w);
                         String myString = NumberFormat.getInstance().format(e);
                         editText2.setText(myString) }
                              else
                               {
                           Toast.makeText(getApplicationContext(), "Record is not available..
Enter valid number", Toast.LENGTH_SHORT).show();
 }}
      catch(JSONException e)
            {
      Log.e("log_tag", "Error parsing data "+e.toString());
                          Toast.makeText(getApplicationContext(), "JsonArray fail",
Toast.LENGTH_SHORT).show();
            }}
                        });
                        buttonupadte.setOnClickListener(new View.OnClickListener()
                  {
                  public void onClick(View view)
                   {
                     String result = null;
                     InputStream is = null;
                     EditText editText = (EditText)findViewById(R.id.e1);
String v1 = editText.getText().toString();
      EditText editText1 = (EditText)findViewById(R.id.e2);
String v2 = editText1.getText().toString();
```

```
        EditText editText2 = (EditText)findViewById(R.id.e3);
String v3 = editText2.getText().toString();
ArrayList<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
nameValuePairs.add(new BasicNameValuePair("f1",v1));
                nameValuePairs.add(new BasicNameValuePair("f2",v2));
                nameValuePairs.add(new BasicNameValuePair("f3",v3));
//http post
                try{
                        HttpClient httpclient = new DefaultHttpClient();
                        HttpPost httppost = new
HttpPost("http://192.168.1.102/FLOWERS/sudha/update.php");
                        httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                        HttpResponse response = httpclient.execute(httppost);
                        HttpEntity entity = response.getEntity();
                        is = entity.getContent();
 Log.e("log_tag", "connection success ");
                        Toast.makeText(getApplicationContext(), "pass",
Toast.LENGTH_SHORT).show();
                }
                catch(Exception e)
                {
                        Log.e("log_tag", "Error in http connection "+e.toString());
                        Toast.makeText(getApplicationContext(), "Connection fail",
Toast.LENGTH_SHORT).show();

                }
                //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");
                         //Toast.makeText(getApplicationContext(), "Record Inserted",
Toast.LENGTH_SHORT).show();
                       //   Intent i = new Intent(getBaseContext(),DatabaseActivity.class);
                      //    startActivity(i);
                    }
                    is.close();

                    result=sb.toString();
               }
               catch(Exception e)
               {
                    Log.e("log_tag", "Error converting result "+e.toString());
                 // Toast.makeText(getApplicationContext(), " record passing fail",
Toast.LENGTH_SHORT).show();

               }
//parse json data
               try{
       JSONObject json_data = new JSONObject(result);
                   CharSequence w= (CharSequence) json_data.get("re");
                   Toast.makeText(getApplicationContext(), w,
Toast.LENGTH_SHORT).show();
}
       catch(JSONException e)
               {
```

```
                            Log.e("log_tag", "Error parsing data "+e.toString());
                            Toast.makeText(getApplicationContext(), "JsonArray fail",
Toast.LENGTH_SHORT).show();
                        }  }
                    });
 }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                    // Inflate the menu; this adds items to the action bar if it is present.
                    getMenuInflater().inflate(R.menu.ia, menu);
                    return true;
          }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                    // Handle action bar item clicks here. The action bar will
                    // automatically handle clicks on the Home/Up button, so long
                    // as you specify a parent activity in AndroidManifest.xml.
                    int id = item.getItemId();
                    if (id == R.id.action_settings) {
                              return true;
                    }
                    return super.onOptionsItemSelected(item);
          }
}
```

//ip

```
package com.sgp;


public class ip {
            public String getIp()
            {
String ip="192.168.173.1";
return ip;}}
```

//MainActivity

```
package com.sgp;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;


public class MainActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.activity_main);
             ImageButton buttonf=(ImageButton)findViewById(R.id.imageButton4);
        buttonf.setOnClickListener(new View.OnClickListener() {
@Override
```

```
public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),FacultyActivity.class);
                        startActivity(i);
                    }
                });
        ImageButton button1=(ImageButton)findViewById(R.id.imageButton1);
        button1.setOnClickListener(new View.OnClickListener() {
@Override
 public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),FacultyActivity.class);
                        startActivity(i);
                    }
                });
        ImageButton button2=(ImageButton)findViewById(R.id.imageButton2);
        button2.setOnClickListener(new View.OnClickListener() {
@Override
 public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),Department.class);
                        startActivity(i);
                    }
                });
         ImageButton button3=(ImageButton)findViewById(R.id.imageButton3);
        button3.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),MarksActivity.class);
                        startActivity(i);
                    }
                });
        ImageButton button4=(ImageButton)findViewById(R.id.imageButton4);
        button4.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),MapsActivity.class);
                        startActivity(i);
                    }
                });
 ImageButton button5=(ImageButton)findViewById(R.id.imageButton5);
button5.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),AboutActivity.class);
                        startActivity(i);
                    }
                });
       ImageButton button6=(ImageButton)findViewById(R.id.imageButton6);
        button6.setOnClickListener(new View.OnClickListener() {
 @Override
public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Intent i = new Intent(getApplicationContext(),EventActivity.class);
                        startActivity(i);
                }
            });
        }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
        }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
        }
}
```

//MapsActivity

```
 package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MapsActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_maps);
                WebView wv = (WebView)findViewById(R.id.my_webview);
     wv.getSettings().setDomStorageEnabled(true);
     wv.getSettings().setBuiltInZoomControls(true);
     wv.getSettings().setSupportZoom(true);
     wv.getSettings().setSupportMultipleWindows(true);
     wv.getSettings().setLoadWithOverviewMode(false);
     wv.getSettings().setJavaScriptEnabled(true);
     wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
    wv.getSettings().setUseWideViewPort(false);
    wv.setWebViewClient(new WebViewClient() {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
          view.loadUrl(url);
          return true;
        }
     });
```

```
        ip ip = new ip();
        String ip_address=ip.getIp();
         wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/maps.jsp");
           }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.maps, menu);
                return true;
           }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
           }
}
```

//MarksActivity

```
package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MarksActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_marks);
            WebView wv = (WebView)findViewById(R.id.my_webview);
    wv.getSettings().setDomStorageEnabled(true);
    wv.getSettings().setBuiltInZoomControls(true);
    wv.getSettings().setSupportZoom(true);
    wv.getSettings().setSupportMultipleWindows(true);
    wv.getSettings().setLoadWithOverviewMode(false);
    wv.getSettings().setJavaScriptEnabled(true);
    wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
    wv.getSettings().setUseWideViewPort(false);
    wv.setWebViewClient(new WebViewClient() {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
      }
    });
```

```
        ip ip = new ip();
        String ip_address=ip.getIp();
wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/internals.jsp");
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.marks, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

//MechActivity

```
 package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MechActivity extends Activity {
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.activity_mech);
                        WebView wv = (WebView)findViewById(R.id.my_webview);
        wv.getSettings().setDomStorageEnabled(true);
        wv.getSettings().setBuiltInZoomControls(true);
        wv.getSettings().setSupportZoom(true);
        wv.getSettings().setSupportMultipleWindows(true);
        wv.getSettings().setLoadWithOverviewMode(false);
        wv.getSettings().setJavaScriptEnabled(true);
        wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
        wv.getSettings().setUseWideViewPort(false);
       wv.setWebViewClient(new WebViewClient() {
@Override
   public boolean shouldOverrideUrlLoading(WebView view, String url) {
               view.loadUrl(url);
               return true;
           }
        });
	ip ip = new ip();
	String ip_address=ip.getIp();
	wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/mech.jsp");
	    }
	    @Override
	    public boolean onCreateOptionsMenu(Menu menu) {
			// Inflate the menu; this adds items to the action bar if it is present.
			getMenuInflater().inflate(R.menu.mech, menu);
			return true;
	    }
	    @Override
	    public boolean onOptionsItemSelected(MenuItem item) {
			// Handle action bar item clicks here. The action bar will
			// automatically handle clicks on the Home/Up button, so long
			// as you specify a parent activity in AndroidManifest.xml.
			int id = item.getItemId();
			if (id == R.id.action_settings) {
				return true;
			}
			return super.onOptionsItemSelected(item);
}
```

//MetActivity

```
package com.sgp;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MetActivity extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_met);
                WebView wv = (WebView)findViewById(R.id.my_webview);
      wv.getSettings().setDomStorageEnabled(true);
      wv.getSettings().setBuiltInZoomControls(true);
      wv.getSettings().setSupportZoom(true);
      wv.getSettings().setSupportMultipleWindows(true);
      wv.getSettings().setLoadWithOverviewMode(false);
      wv.getSettings().setJavaScriptEnabled(true);
      wv.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
      wv.getSettings().setUseWideViewPort(false);
      wv.setWebViewClient(new WebViewClient() {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
           view.loadUrl(url);
           return true;
        }
      });
```

```
        ip ip = new ip();
        String ip_address=ip.getIp();
        wv.loadUrl("http://"+ip_address+":8089/SGP_WEB/mobile/met.jsp");
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.met, menu);
            return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action bar will
            // automatically handle clicks on the Home/Up button, so long
            // as you specify a parent activity in AndroidManifest.xml.
            int id = item.getItemId();
            if (id == R.id.action_settings) {
                    return true;
            }
            return super.onOptionsItemSelected(item);
    }
}
```

//Welcome

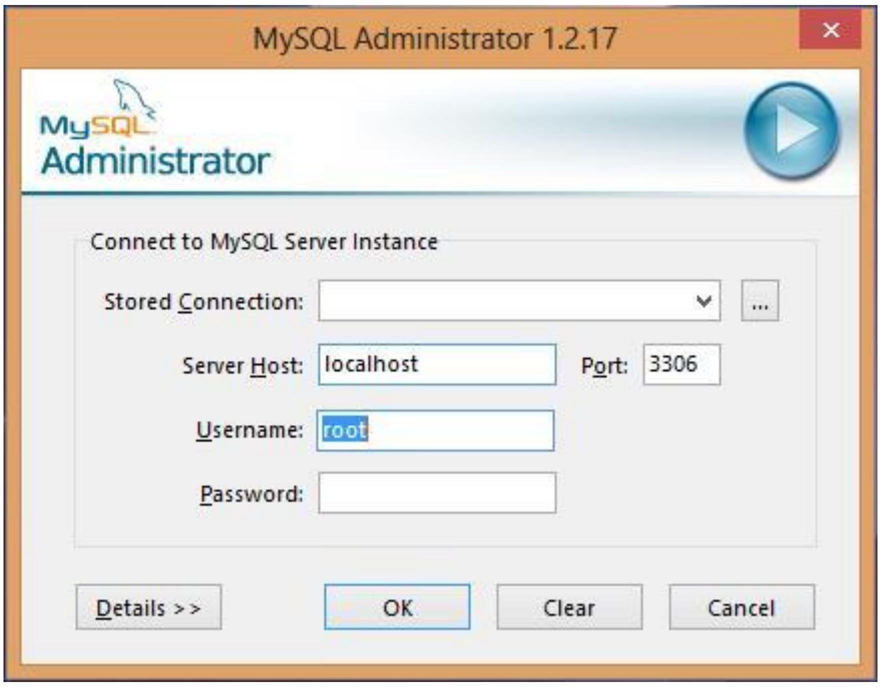
```
 package com.sgp;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
public class Welcome extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_welcome);
		 Button button=(Button)findViewById(R.id.button1);
	    button.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
			   // TODO Auto-generated method stub
			   Intent i = new Intent(getApplicationContext(),MainActivity.class);
			   startActivity(i);
	  } });}
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.welcome, menu);
		return true;}
	@Override
```

```
        public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
}
}
```

CHAPTER 7

DATABASE DESIGN

7.1 Getting started with MySQL:



Stored Connection:

Server Host: localhost Port: 3306

Username: root

Password:

Fig: 7.1 Connect To Mysql Server

7.2 Normalization :

A Database is a collection of interrelated data stored with a minimum of redundancy to serve many applications. The database design is used to group data into a number of tables and minimizes the artificiality embedded in using separate files. The tables are organized to

1. *Reduced duplication of data.*
2. *Simplify functions like adding, deleting, modifying data etc..,*
3. *Retrieving data*
4. *Clarity and ease of use*
5. *More information at low cost*

Normalization is built around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints on the kind of functional dependencies that could be associated with the relation. The normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced into the database.

First Normal Form:

A relation R is in first normal form if and only if all underlying domains contained atomic values only.

Second Normal Form:

A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

Third Normal Form:

A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

7.3 SGP Database :

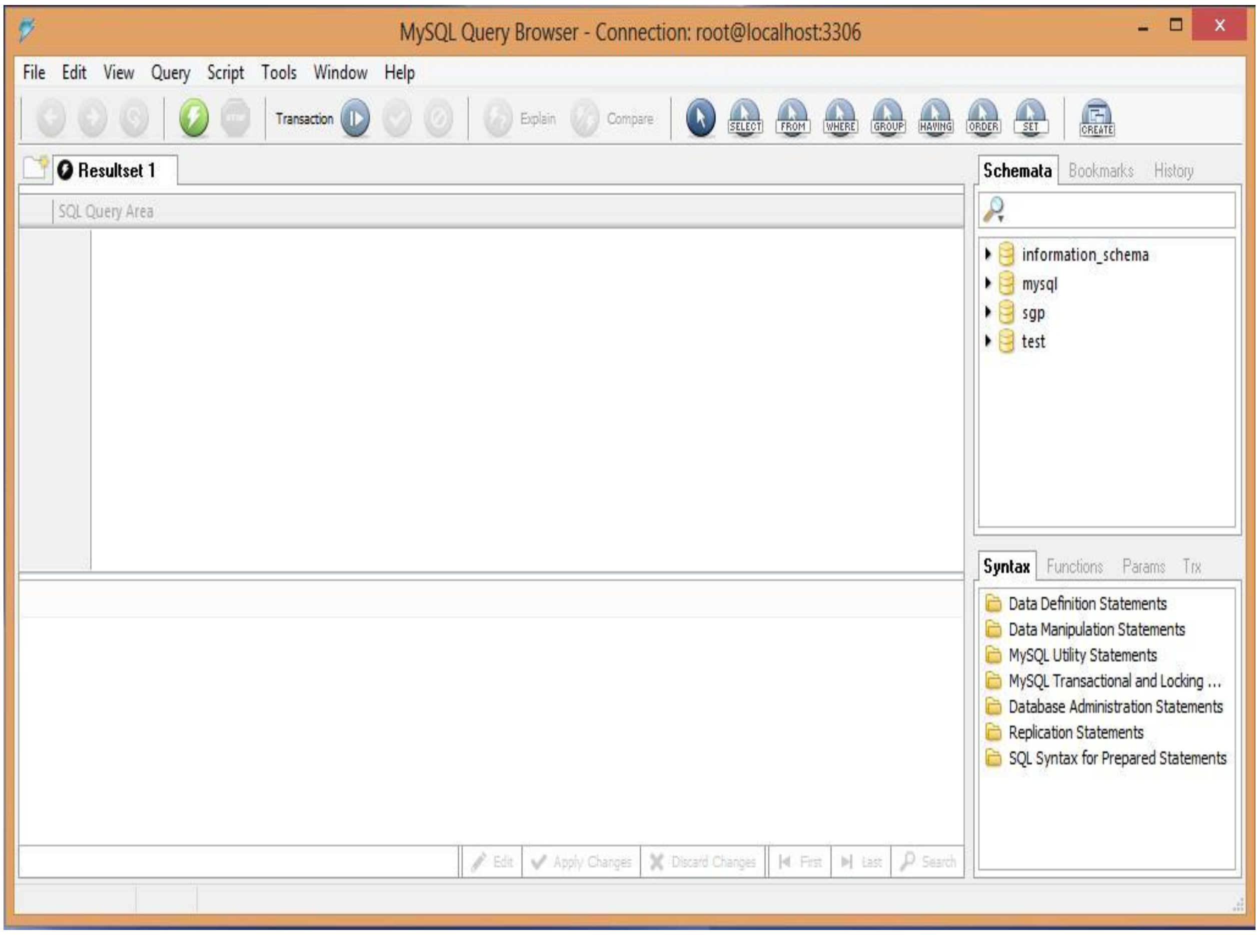
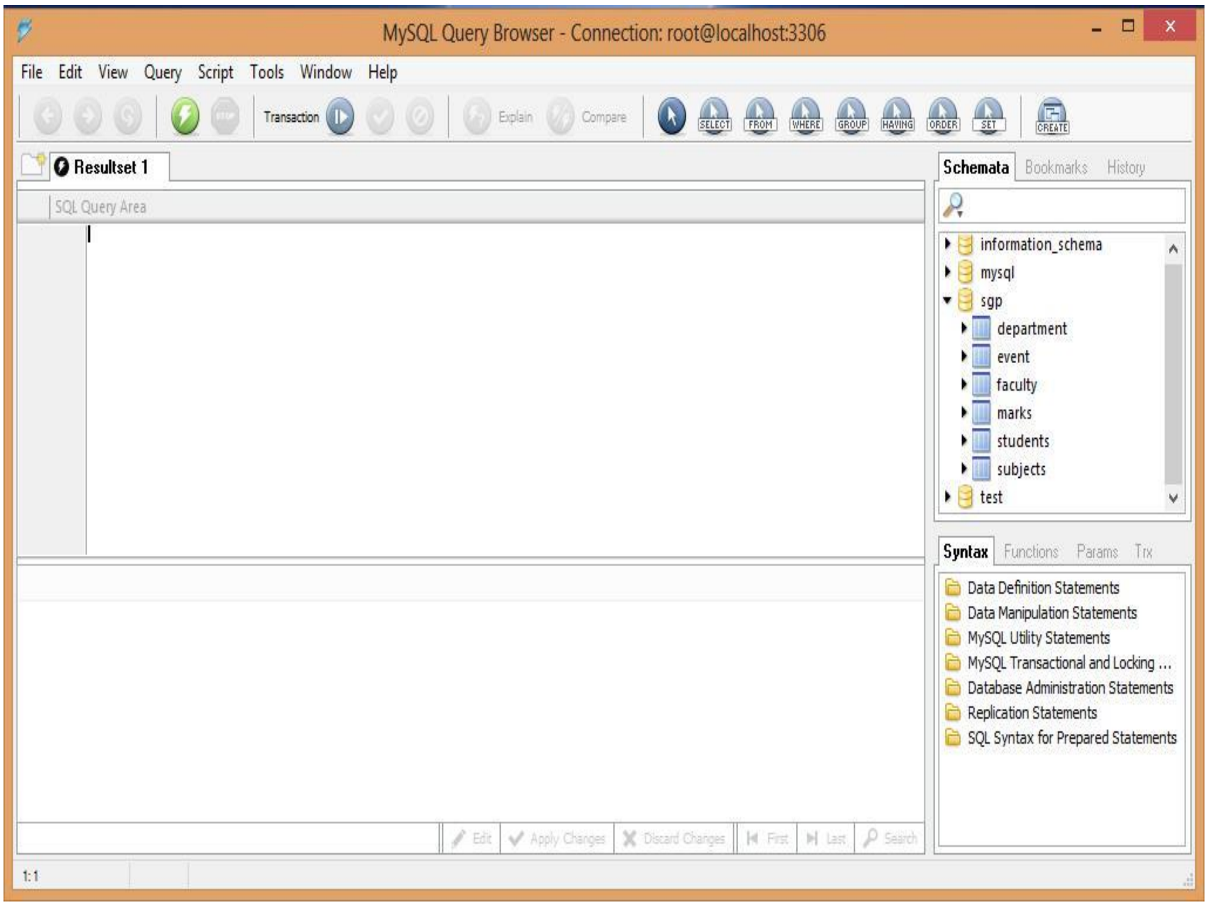

MySQL Query Browser - Connection: root@localhost:3306
File Edit View Query Script Tools Window Help
Transaction
Explain
Compare
SELECT FROM WHERE GROUP HAVING ORDER SET
CREATE
Resultset 1
SQL Query Area
Schemata Bookmarks History
information_schema
mysql
sgp
test
Syntax Functions Params Trx
Data Definition Statements
Data Manipulation Statements
MySQL Utility Statements
MySQL Transactional and Locking ...
Database Administration Statements
Replication Statements
SQL Syntax for Prepared Statements
Edit Apply Changes Discard Changes First Last Search

Fig: 7.2 SGP Database

7.4 SGP Tables :



information_schema
mysql
sgp
department
event
faculty
marks
students
subjects
test

Data Definition Statements
Data Manipulation Statements
MySQL Utility Statements
MySQL Transactional and Locking ...
Database Administration Statements
Replication Statements
SQL Syntax for Prepared Statements

Fig: 7.3 SGP Tables

CHAPTER 7 TESTING

7.1 Introduction to Testing :

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

7.2 Testing In Strategies:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are

7.3 Unit Testing :

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two Strategies:

7.3.1 Black Box Testing :

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

1. Incorrect or missing functions

2. Interface errors

3. Errors in data structure or external database access

4. Performance errors

5. Initialization and termination errors.

In this testing only the output is checked for correctness.

The logical flow of the data is not checked.

7.3.2 White Box testing :

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. *Guarantee that all independent paths have been executed.*
2. *Execute all logical decisions on their true and false Sides.*
3. *Execute all loops at their boundaries and within their operational bounds*
4. *Execute internal data structures to ensure their validity.*

7.4 Integrating Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

7.5 System Testing

It involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

7.6 Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

7.7 Test Approach:

Testing can be done in two ways:

1. *Bottom up approach*
2. *Top down approach*

1. Bottom up Approach :

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

2. Top down approach :

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

7.8 Validation :

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

CHAPTER 9

SNAPSHOTS

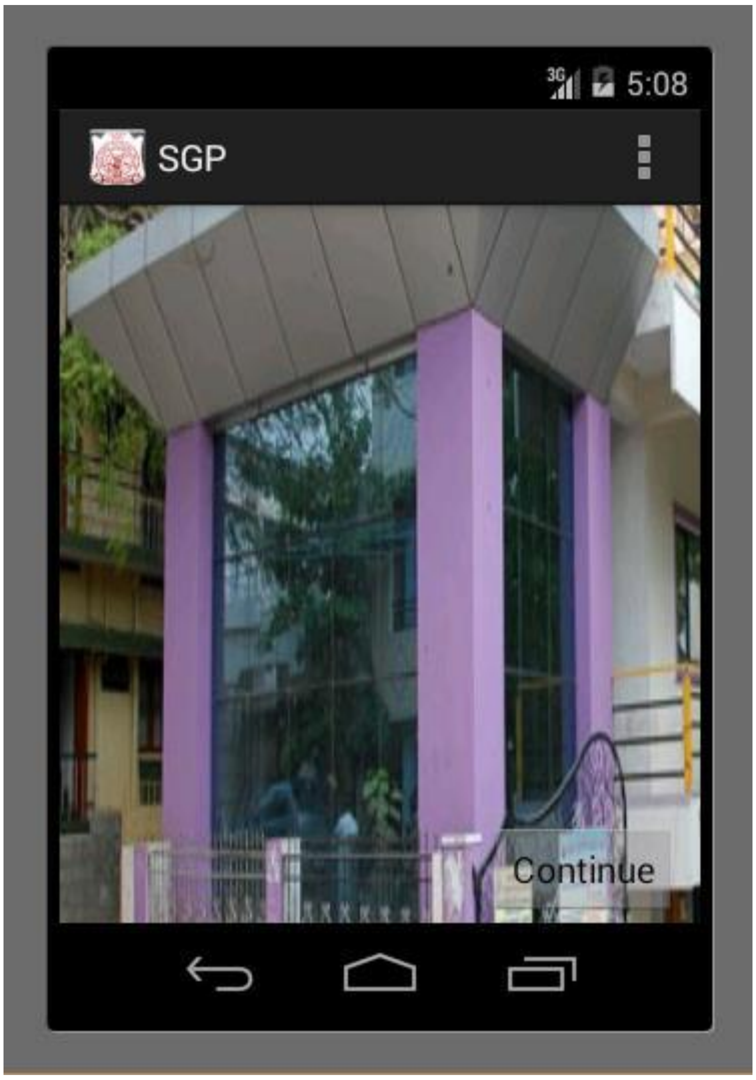


Figure 9.1 Starting Page

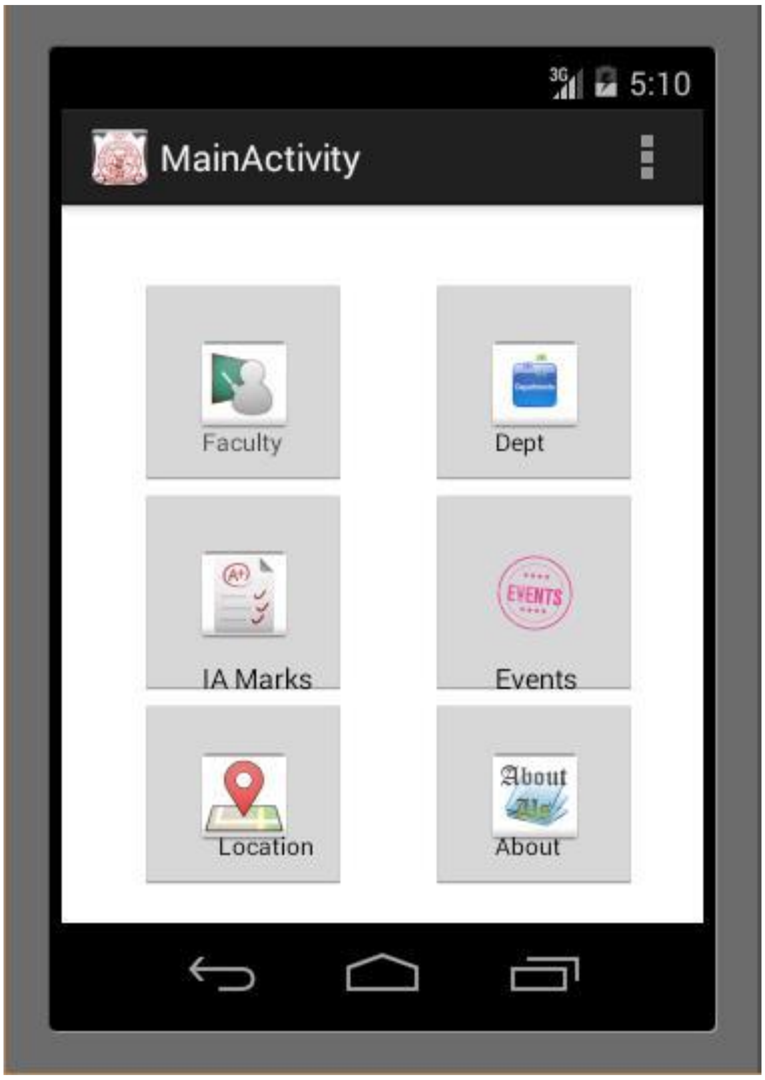


Figure 9.2 Menu

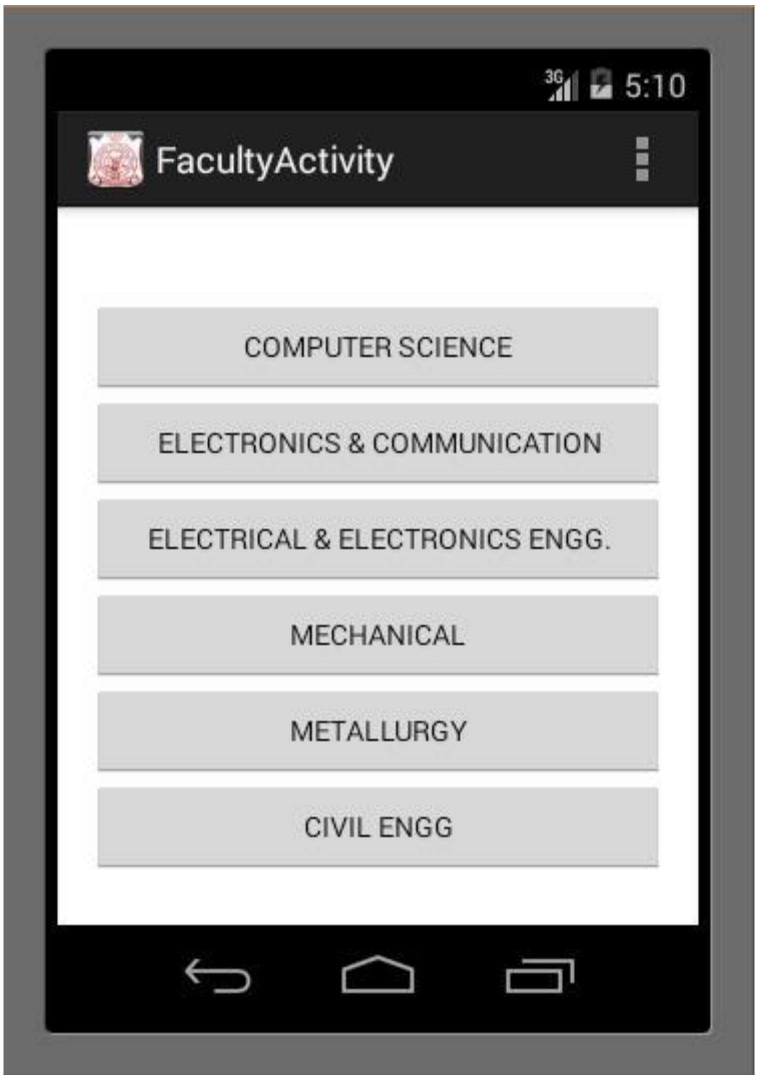
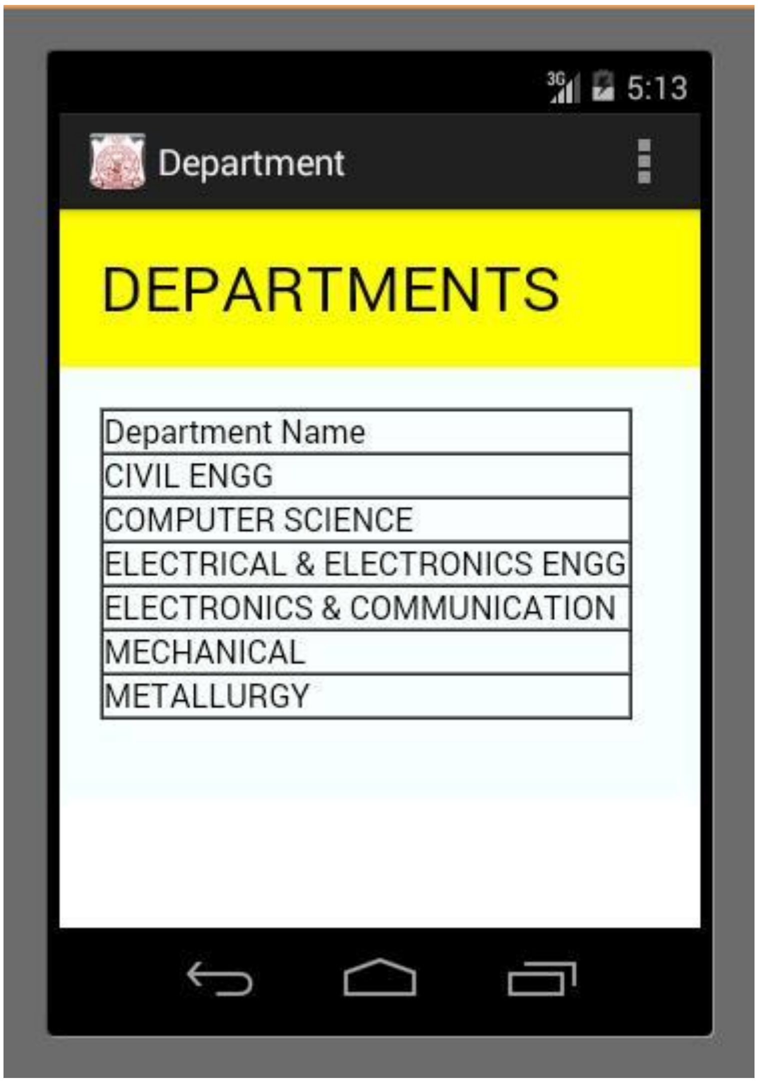


Figure 9.3 Faculty Sub menu



DEPARTMENTS

Department Name
CIVIL ENGG
COMPUTER SCIENCE
ELECTRICAL & ELECTRONICS ENGG
ELECTRONICS & COMMUNICATION
MECHANICAL
METALLURGY

Figure 9.4 Departments Module

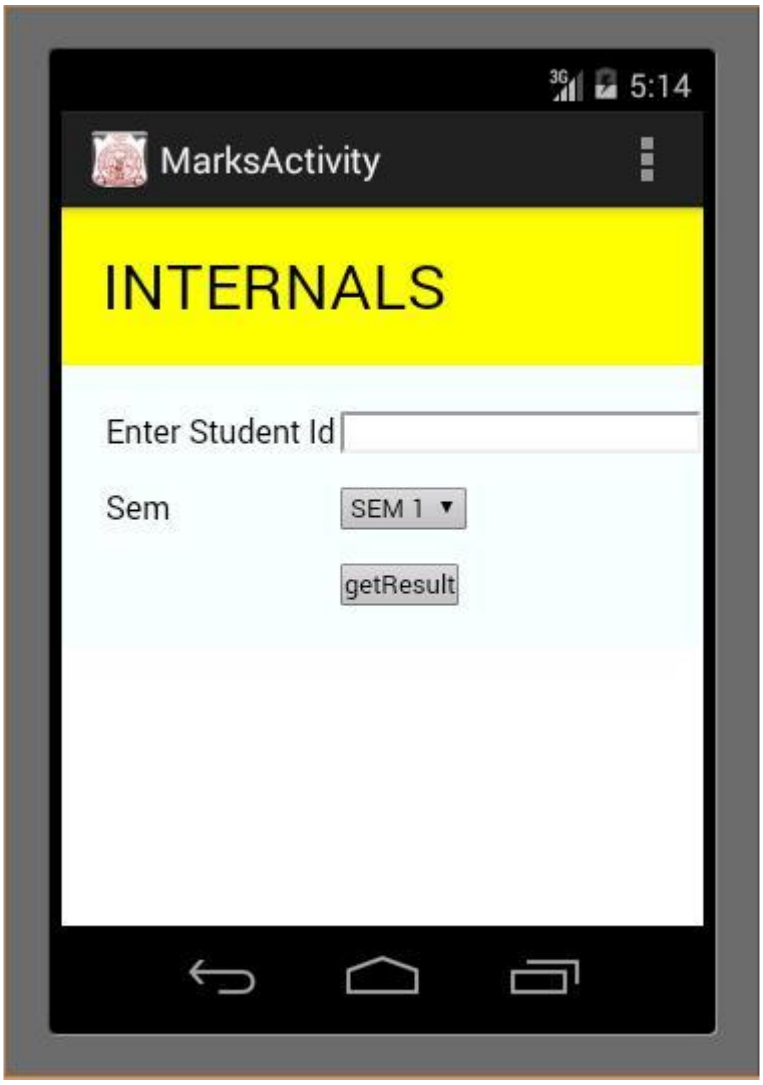
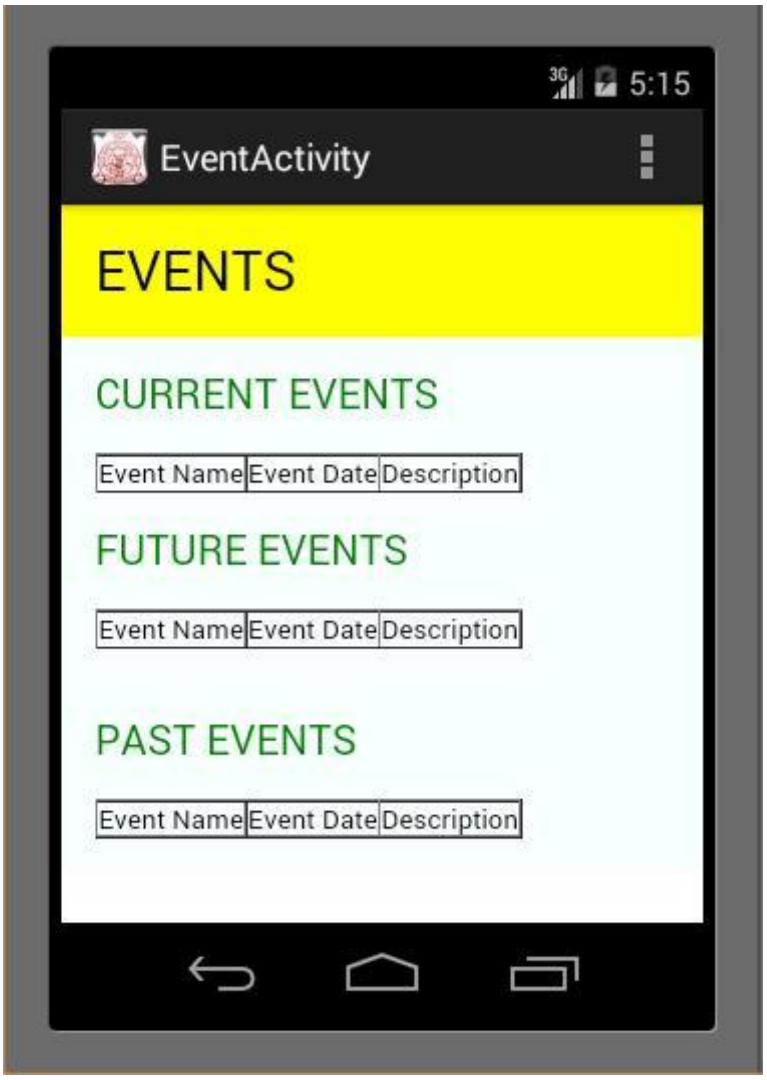


Figure 9.5 Internal Module


5:15

EventActivity

EVENTS

CURRENT EVENTS

Event Name	Event Date	Description

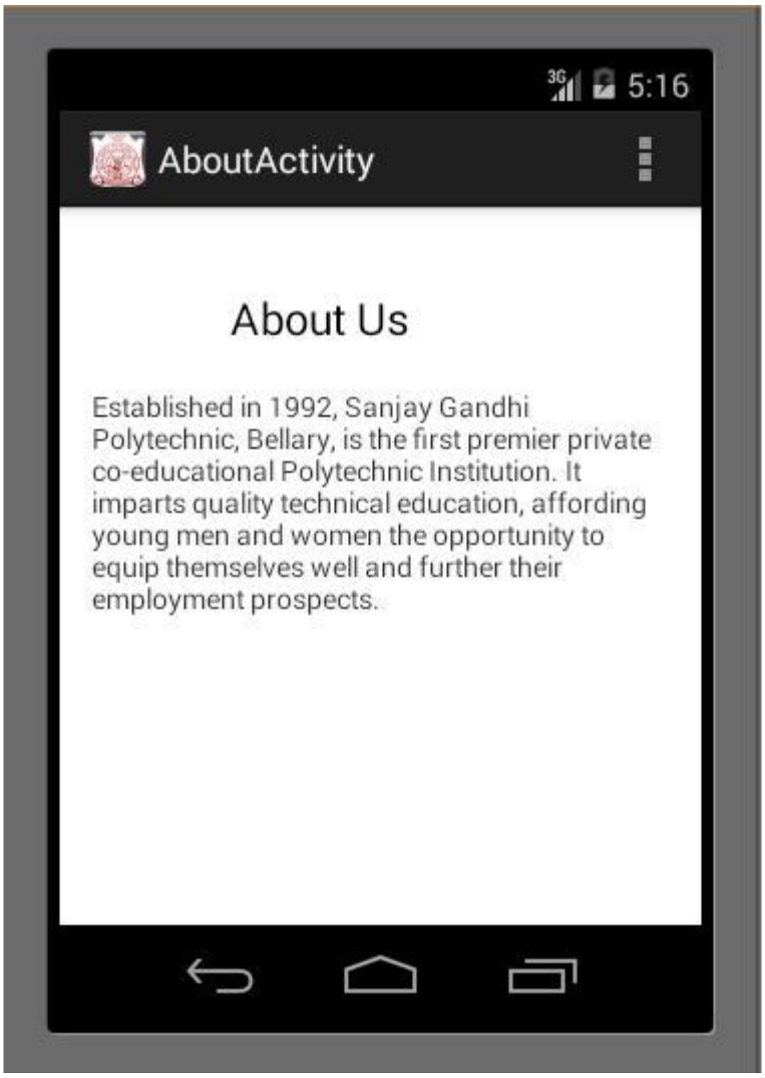
FUTURE EVENTS

Event Name	Event Date	Description

PAST EVENTS

Event Name	Event Date	Description

Figure 9.6 Events Module



About Us

Established in 1992, Sanjay Gandhi Polytechnic, Bellary, is the first premier private co-educational Polytechnic Institution. It imparts quality technical education, affording young men and women the opportunity to equip themselves well and further their employment prospects.

Figure 9.7 About us module

9. CONCLUSION

*I have successfully completed my "**COLLEGE ANDROID APP**" project and achieved the mentioned goals and objectives. The College Android App project is an implementation of a general our college websites, which helps the parents and students to get the information like IA Marks and availability of staff in each departments and their details.*

10. FUTURE ENHANCEMENTS

It is not possible to develop a system that makes all the requirements of the users. User requirements keep changing as the system is being used. Some of the

future enhancements that can be done to this system are:

1. As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
2. Because it is based on object-oriented design, any further changes can be easily adaptable.
3. Based on the future security issues, security can be improved using emerging technologies.
.

11. BIBLIOGRAPHY

I. **www.tutorialspoint.com**

II. ***Herbert Schildt, The Complete Reference-JAVA.***

III. ***cay s.Hortsman, core java 2 volume II-advanced.***

IV. ***Gray Cornell,Pearson Education-Sun Microsystems.***

V. ***Herbert Schildt, The Complete Reference-JAVA.***

VI. ***www.wikipedia.org***

12. USER MANUAL

Steps to run the COLLEGE ANDROID APP on PC:

1. Install jdk1.6.

2. Install Eclipse tool

3. Install MySQL.

4. install Tomcat Server.

5. Download the Android SDK packages.

5. Load the project into the eclipse and run the project.

6. Run the MySql and Tomcat Server.