# 1 INTRODUCTION

**PROBLEM DEFINITION**

* Mind Tech Bug and Component System is a tool is used to keep track of the bugs occurred among the several projects. Lot of effort was put to make it user friendly.

Optimum utilization of tool is possible. All basic features are provided. In every organization, every product that needs to be tested before making it available to the society. So, we have to maintain the details of test process. In the present project **“Mind Tech Bug Component System”** we are able to keep track all the testing details of bugs occurred in different projects. In the standalone system, it is some more difficult to maintain different databases on every system. So in every branch we have to install the software and maintain a database individually for tracking the bugs occurred in that branch.

2.SYSTEM SPECIFICATIONS

**2.1 SOFTWARE REQUIREMENTS:**

Language & technology : Android SDK

IDE : Eclipse

Operating system : Windows XP and above

**2.2 HARDWARE REQUIREMENTS:**

Processor : (min) p4 processor.

RAM : 1GB

Hard Disk : 20GB

3. SYSTEM ANALYSIS

**3.1 Existing System:**

The existing system is also computerized one. But it is standalone computer system application. So in every branch we have to install the software and maintain a database individually for tracking the bugs occurred in that branch. At final it is very difficult to keep track of all bugs occurred around all the branches. Totally is very time taking and risky process to collect all the bugs from every branch and controlling them and taking the decisions on those collected data about the bugs.

**3.2 Proposed System:**

The first step of analysis process involves the identification of need. The success of a system depends largely on how accurately a problem is defined, thoroughly investigated and properly carried out through the choice of solution.This application has been developed in order to overcome the difficulties encountered while using the existing system. This is web-based distributed JSP Bug. An integration of JSP/Java-Servlets, and relational database.

**3.3 Feasibility Study:**

The feasibility of the project is analyzed in this phase and business

proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

TECHNICAL FEASIBILITY

SOCIAL FEASIBILITY

**3.4 MODULE DESCRIPTION**

In the Project titled “MindTech Bug Component System”, the modules are:

1.Super User

2.Administrator

3.Developer

4.Tester

**Super User**: The Application is designed to keep track of bugs that are raised during the Life Cycle of Softwaredevelopment. The Super User maintains the details of Employees along with department, Projects with Components and Versions. Super user can add Employees for different projects. After logon the Super User, he can enter the Super User homepage and from there he can do the activities like adding users in the organization.

**Administrator**: Administrators can add Projects to this application. The Bugs raised will be reported to the Administrator or Super User. The Bug will be assigned to different Users. The Bugs will be maintained under different conditions with different priorities and Status. Nearly administrator is also having same rights like Super User except addition of users.

**Developer:** Developer can see the bugs in his project. Others cannot view the Bugs assigned to a particular employee.

**Tester:**The tester can report the bugs to the administrators. He can send the bugs severity, priority of bugs, and the bug description to the administrator either through the interface provided or through mail.

**4 System Design:**

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as in a stylized software engineering process." Software design usually involves problem solving and planning a software solution. This includes both a low-level component design and a high-level, architecture 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.

SMS

User

Android SDK

SQLite

Java program

Get

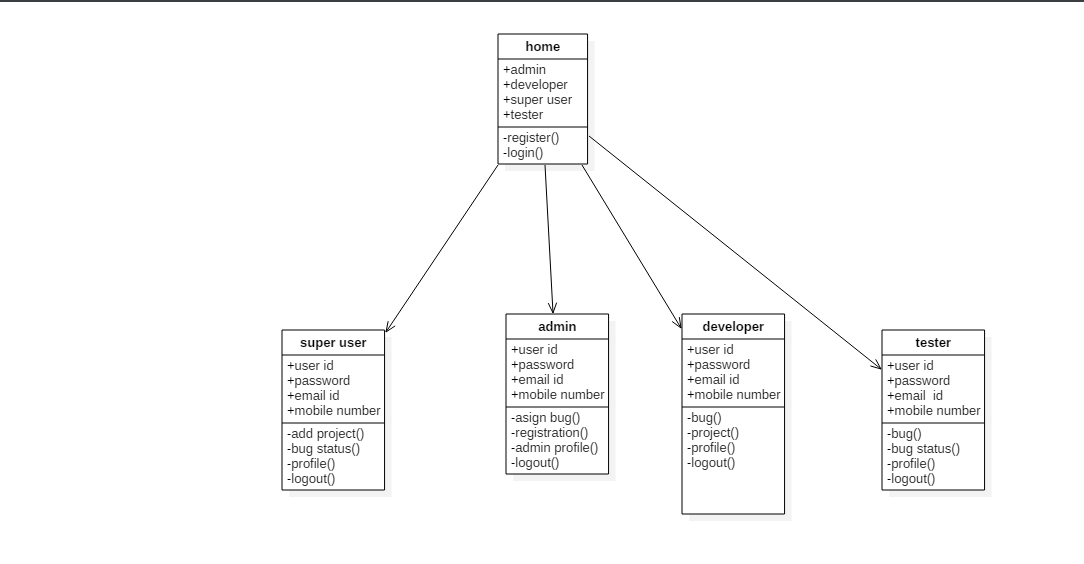
CSV file

**Fig 6.2 Architecture**

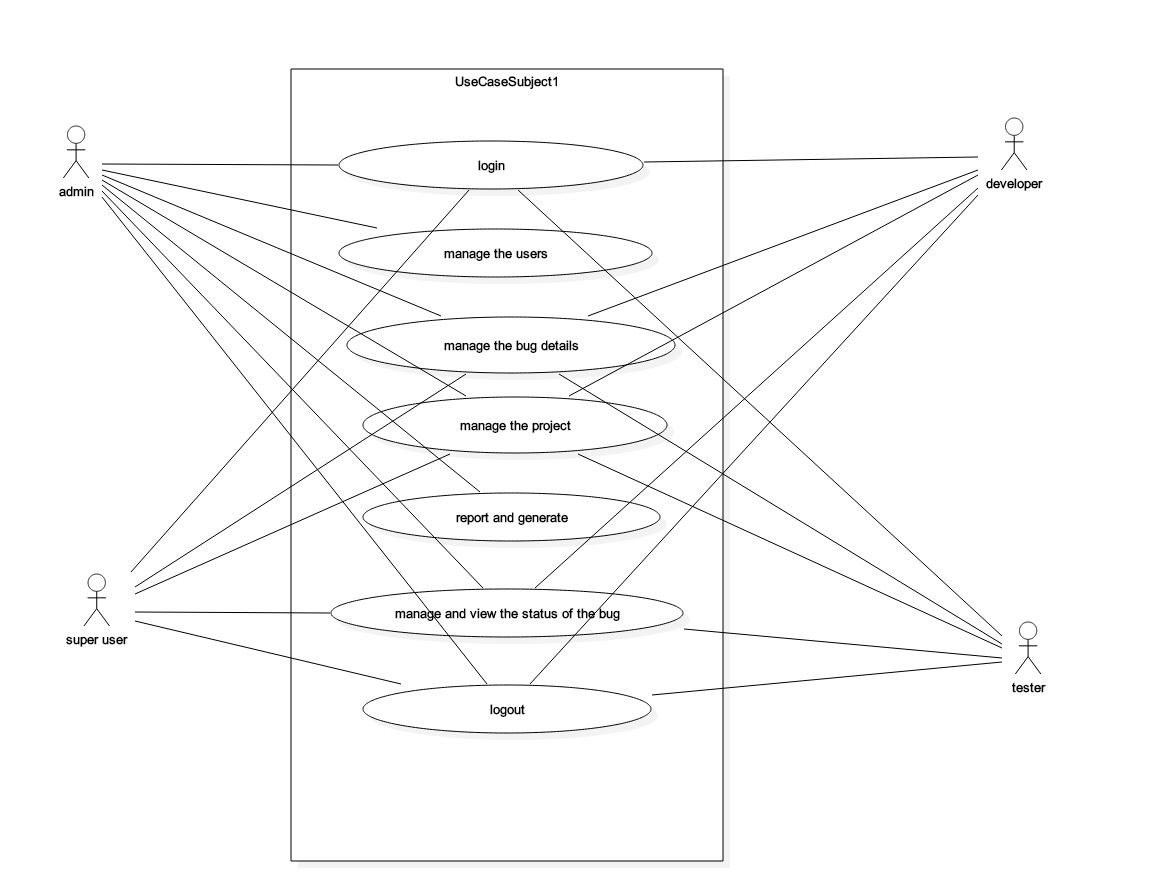
A SMS user for who the application looks like an user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and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

**4.2 UML Diagrams:**

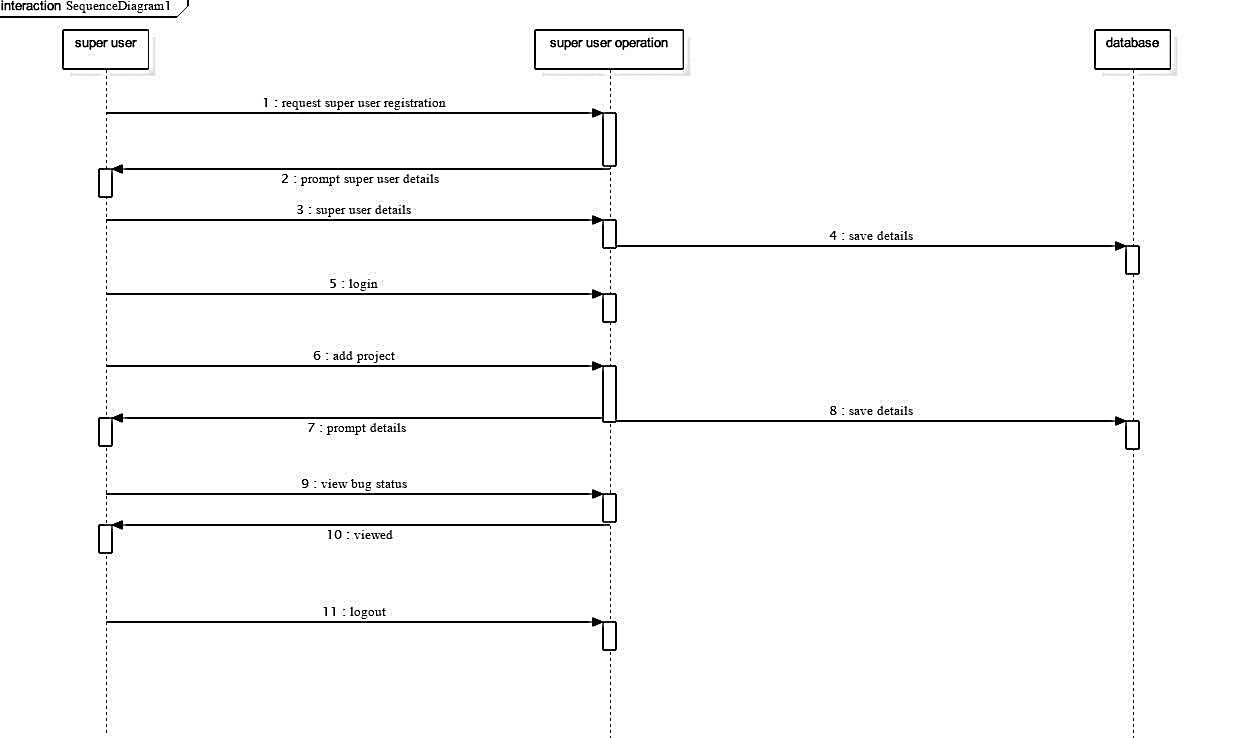
**4.2.1 class diagram:**

****

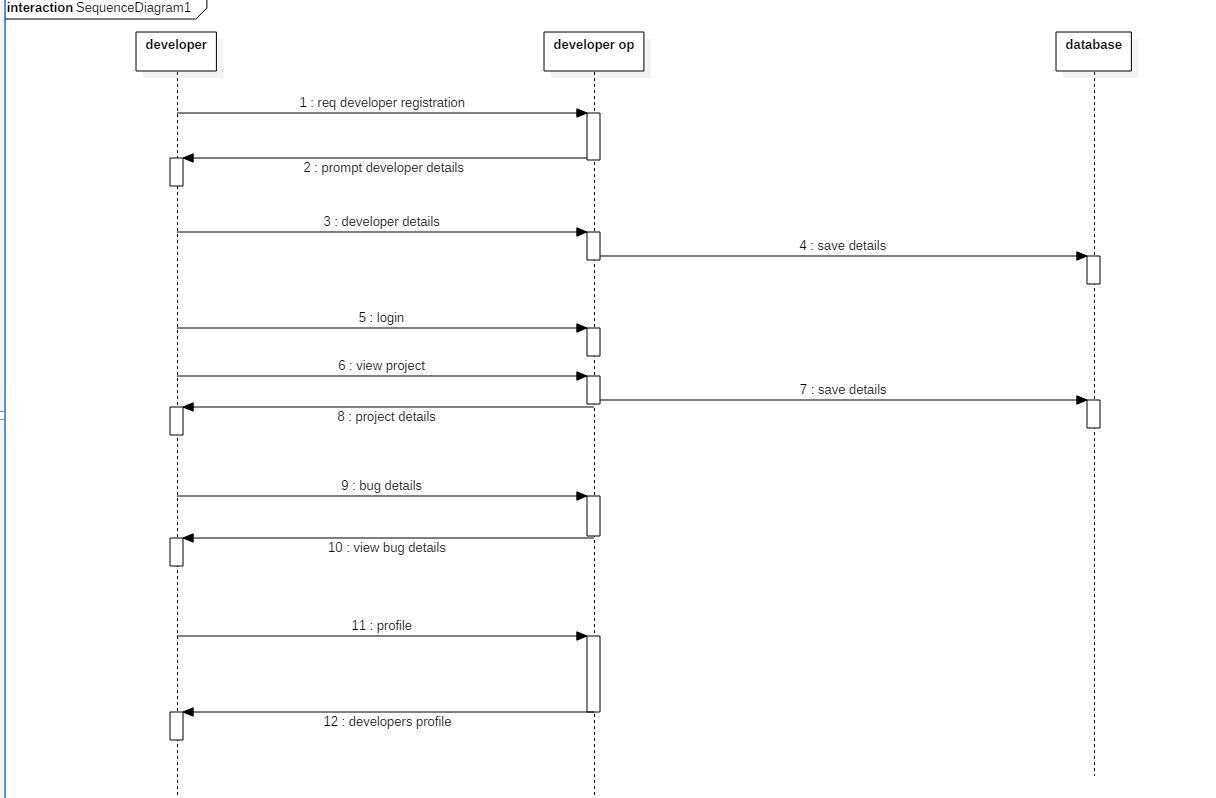
**4.2.2 use case diagram:**

****

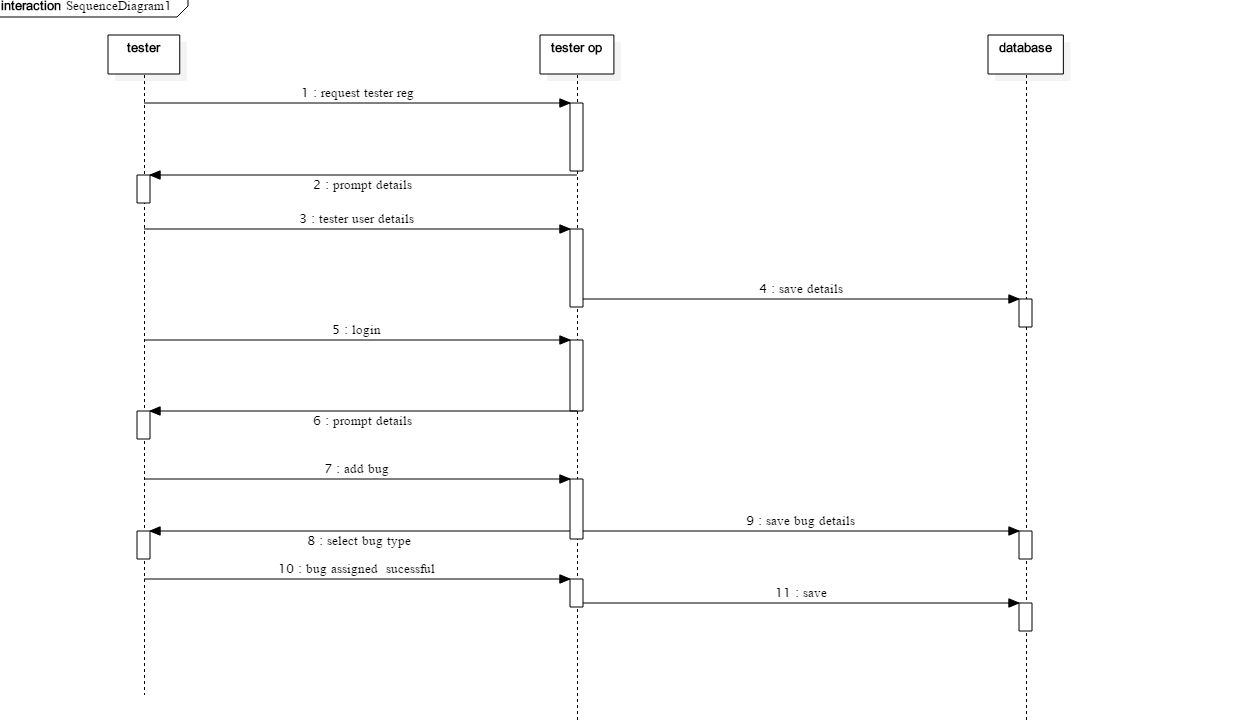
4.2.3 Sequence for super user:

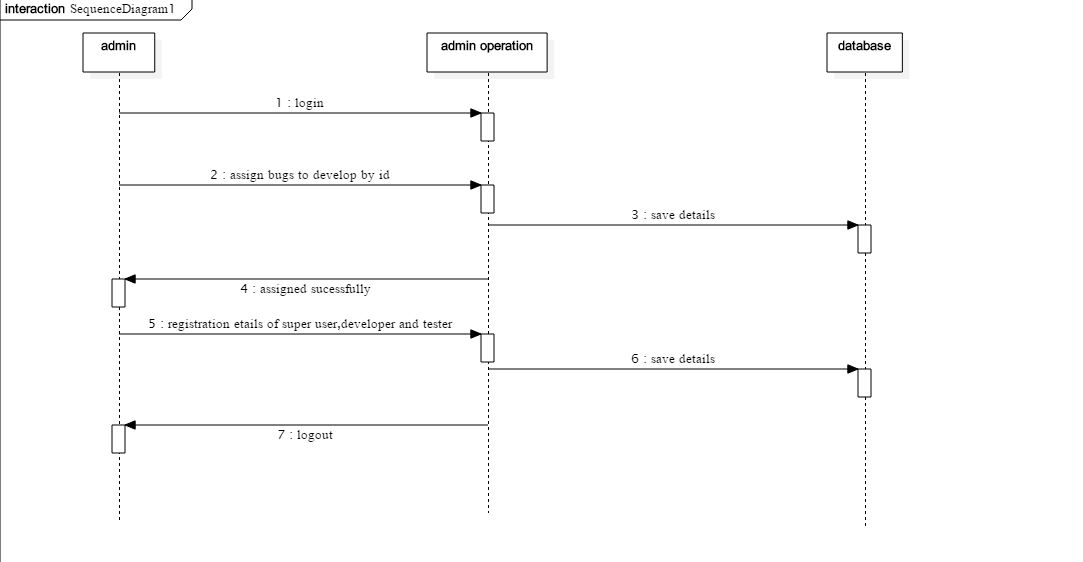


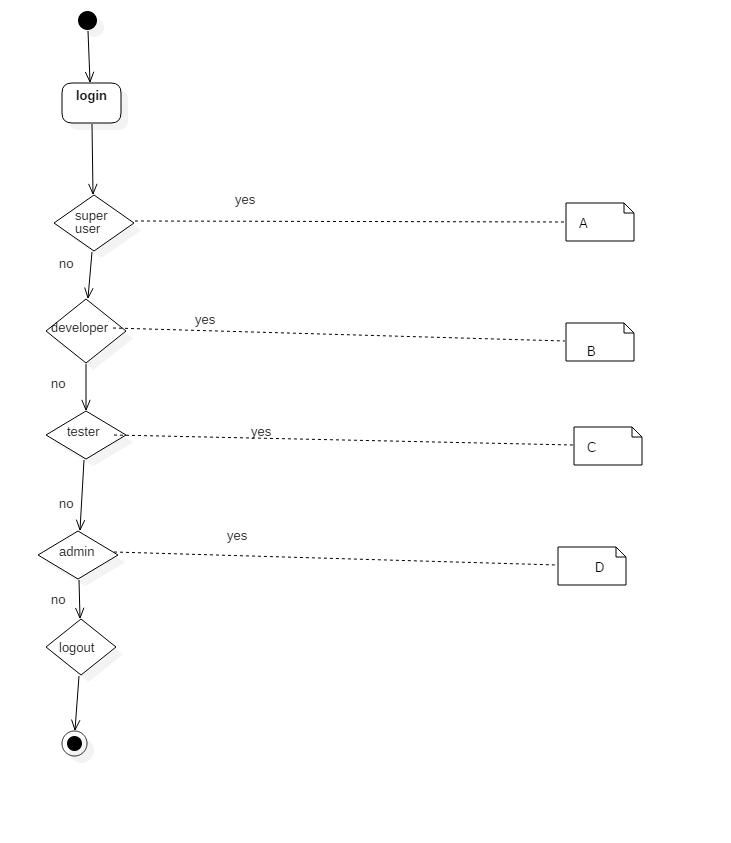
**Sequence of developer:**



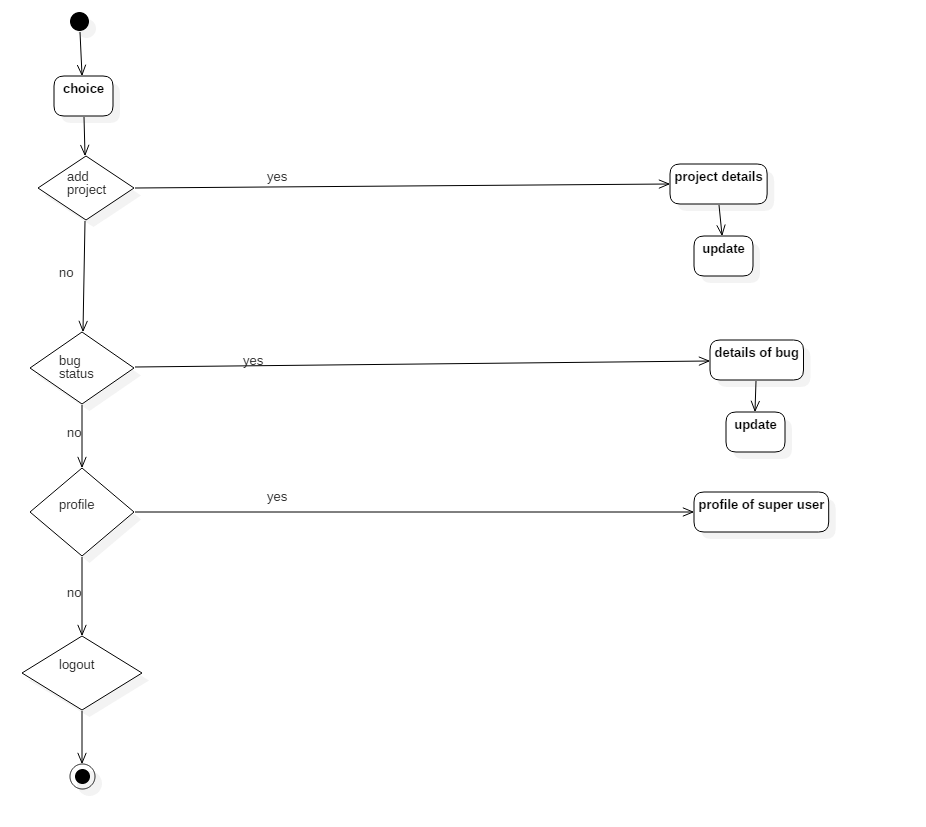
**Sequence of tester:**



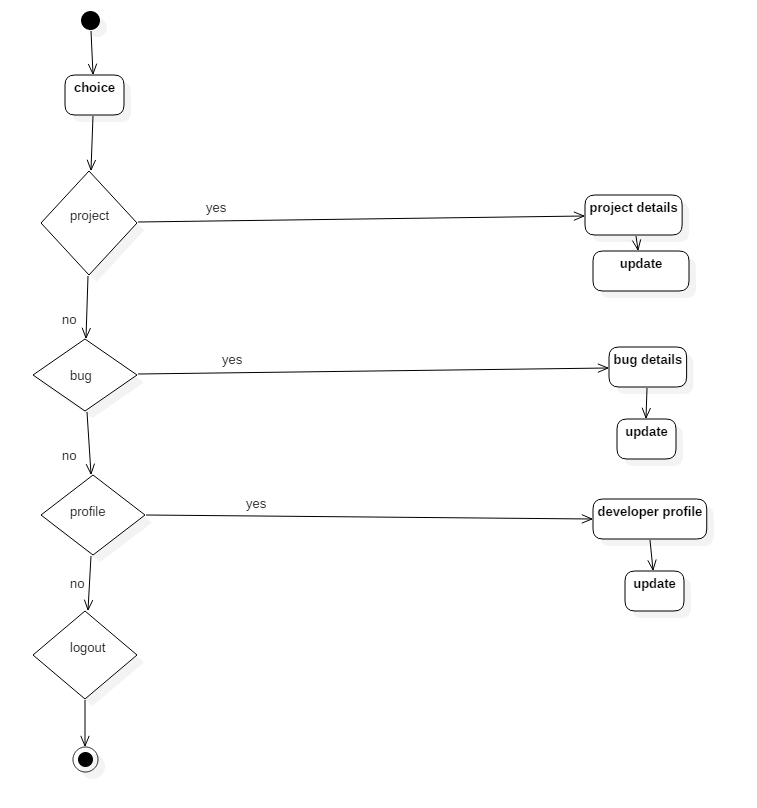
Sequence of admin 

4.2.4 ACTIVITY 

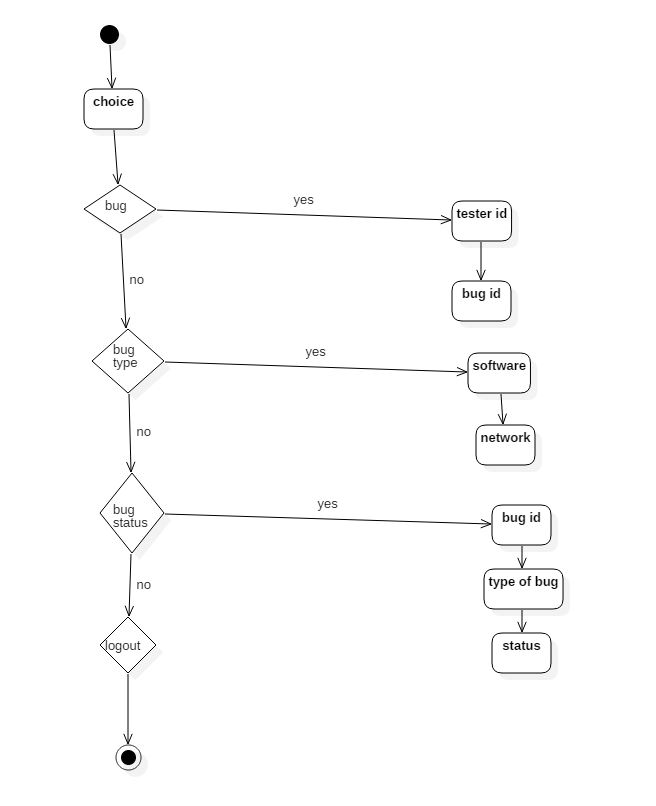
Activity of super user:



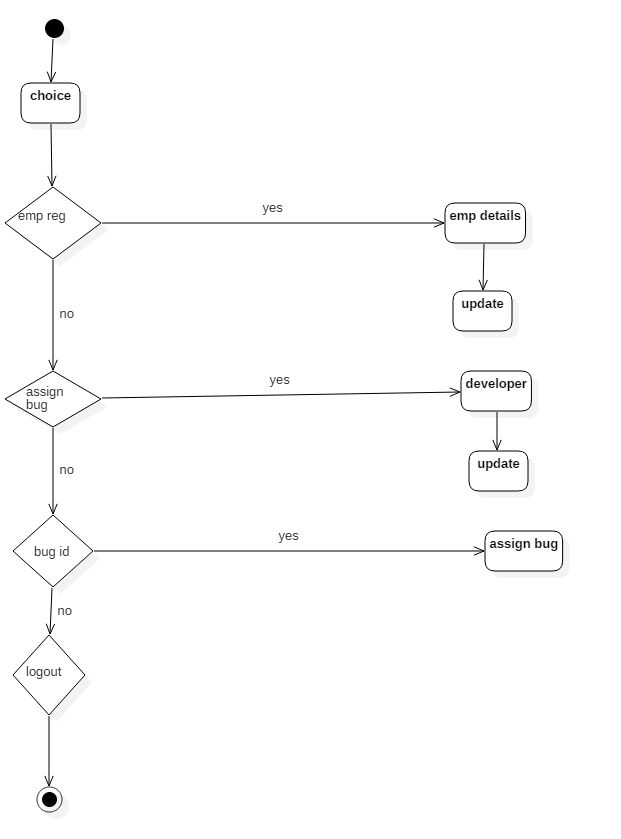
Activity of developer:



Activity of tester:



Activity of admin:



**5. TESTING AND DEBUGGING**

**PURPOSE**

The purpose of testing is to assess product quality. It helps to strengthen and stabilize the architecture early in the development cycle. We can verify through testing, the various interactions, integration of components and the requirements which were implemented. It provides timely feedback to resolve the quality issues, in a timely and cost effective manner. The test workflow involves the following:

* Verifying the interactions of components.
* Verifying the proper integration of components.
* Verifying that all requirements have been implemented correctly.
* Identifying and ensuring that all discovered defects are addressed before the software is deployed.

**QUALITY**

The common usage of the term quality refers to a number of things: principally it means the absence of defects, but more importantly, a fitness for a desired purpose. The ultimate goal of testing is to assess the quality of the end product. Quality assessments often consider process quality and organizational factors as well as direct product quality.

**PRODUCT QUALITY**

The role of testing is not to assure quality, but to assess it, and to provide timely feedback so that quality issues can be resolved in a timely and cost-effective manner.

**TESTING IN THE ITERATIVE LIFECYCLE**

Testing is not a single activity, nor is it a phase in the project during which we assess quality. If developers are to obtain timely feedback on evolving product quality, testing must occur throughout the lifecycle: we can test the broad functionality of early prototypes: we can test the stability, coverage and performance of the architecture while there is still an opportunity to fix it; and we can test the final product to assess its readiness for delivery to customers.

**DIMENSIONS OF TESTING**

To assess product quality, different kinds of tests, each one with a different focus, are needed. These tests can be categorized by several dimensions:

* **Quality dimension**: The major quality characteristic or attribute that is the focus of test.
* **Stage of testing**: The point in the lifecycle at which the test, usually limited to a single quality dimension.
* **Type of testing**: The specific test objective for an individual test, usually limited to a single quality dimension.

**STAGES OF TESTING**

Testing is not a single activity, executed all at once. Testing is executed against different types of targets in different stages of the software development. Test stages progress from testing small elements of the system, such as components (unit testing), to testing completed systems (system testing). The four stages have the following purposes:

* + - * **Unit test**: The smallest testable elements of the system are tested individually, typically at the same time those elements are implemented.
      * **Integration test**: The integrated units (or components or subsystems) are tested.
      * System test: the complete application and system (one or more applications) are tested.
      * **Acceptance test**: The complete application (or system) is tested by end users (or representatives) for the purpose of determining readiness for deployment.

These stages occur throughout the lifecycle, with varying emphasis. An early conceptual prototype user in the inception phase to assess the viability of the product vision will be subjected to acceptance tests. Architectural prototype developed during the elaboration phase be subjected to integration and system tests to validate architectural integrity and performance of key architectural elements.

**TYPES OF TESTING**

After a test plan has been developed, system testing begins by testing program modules separately, followed by testing “bundled” modules as a unit. A program module may function perfectly in isolation but fail when interfaced with other modules. The approach is to test each entity with successively larger ones, up to the system test level.

System testing consists of the following steps:

* + - * Program(s) testing.
      * String testing.
      * System testing.
      * User acceptance testing.

**PROGRAM TESTING:**

A program represents the logical elements of system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. Achieving an error-free program is the responsibility of the programmer. Program testing checks for two types of errors: syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax errors. These errors are shown through error messages generated by syntax errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields, out-of range items, and invalid combinations. Since diagnostics do not detect logic errors, the programmer must examine the output carefully for them.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of instructions must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points. The idea is to compare program values against desk-calculated values to isolate the problem.

**STRING TESTING:**

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each portion of the system is tested against the entire module with both test and live data before the entire system is ready to be tested.

**SYSTEM TESTING:**

System testing is designed to uncover weaknesses that were not found in earlier tests. This includes forced system failure and validation of the total system as it will be implemented by its user(s) in the operational environment. Generally, it begins with low volumes of transactions based on live data. The volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. After the candidate system passes the test, the old system is discontinued.

**USER ACCEPTANCE TESTING:**

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system’s procedures operate to system specifications and that the integrity of vital data is maintained. Performance of an acceptance test is actually the user’s show. User motivation and knowledge are critical for the successful performance of the system. Then a comprehensive test report is prepared. The report indicates the system’s tolerance, performance range, error rate, and accuracy.

**SYSTEM DOCUMENTATION:**

All design and test documentation should be finalized and entered in the library for future reference. The library is the central location for maintenance of the new system. The format, organization, and language of each documentation should be in line with system standards.

Test cases:

Module Name: Admin

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO | TEST CASE | INPUT | EXPECTED  OUTPUT | ACTUAL OUTPUT | PASS/FAIL |
| 1 | Check admin login functionality | Admin username and password | Login must be successful | Login success | Pass |
| 2 | Assign Bugs | Detail Regarding bugs | Must be added successfully | Added successfully | Pass |

Module Name: SuperUser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO | TEST CASE | INPUT | EXPECTED  OUTPUT | ACTUAL  OUTPUT | PASS/FAIL |
| 1 | Check Superuser login functionality | Superuser Username and Password | Login must be successful | Login success | Pass |
| 2 | Add Project | Enter Project details | Must be project details successfully | Enter details successfully | Pass |

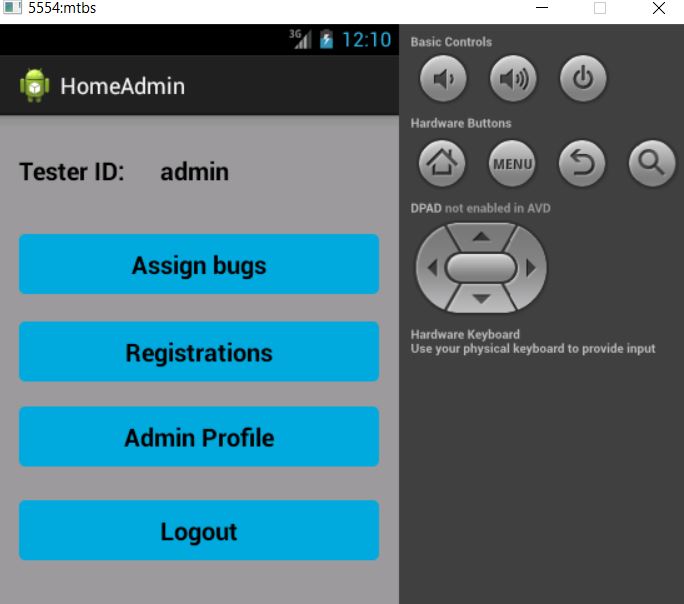
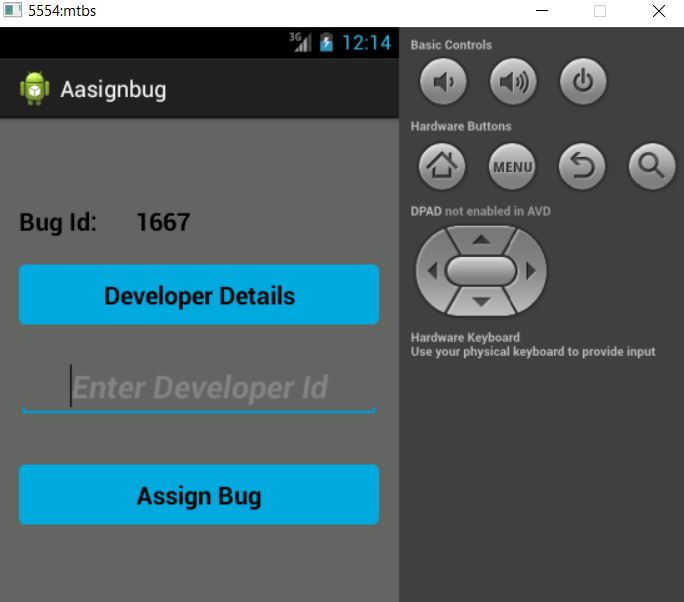
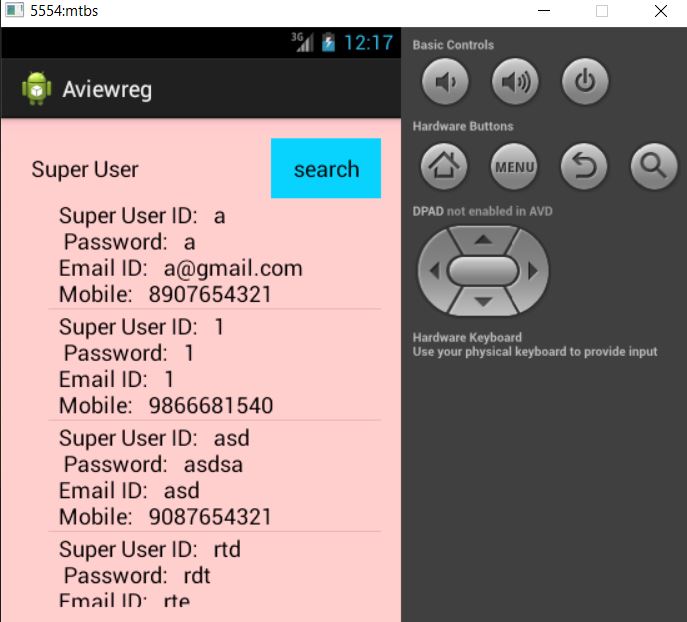
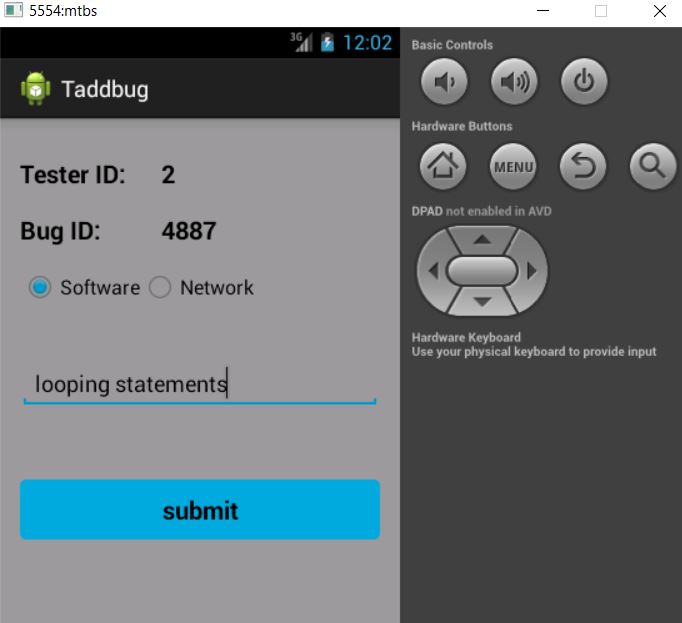
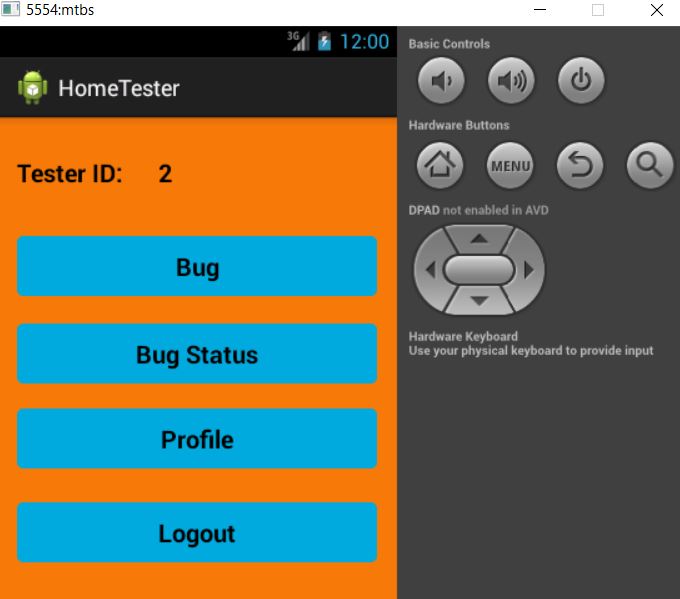
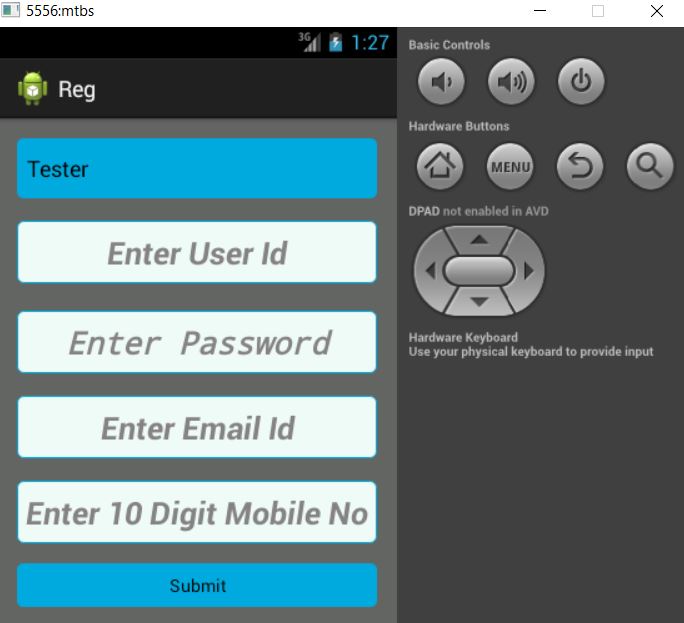
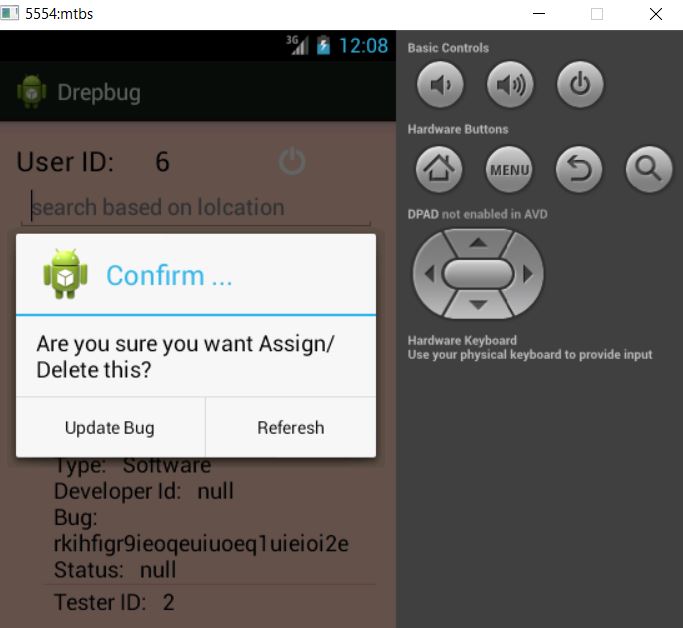
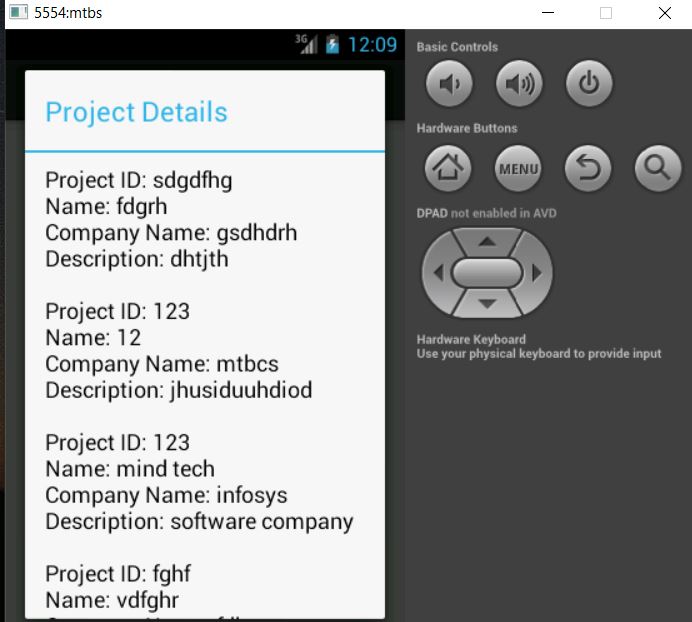
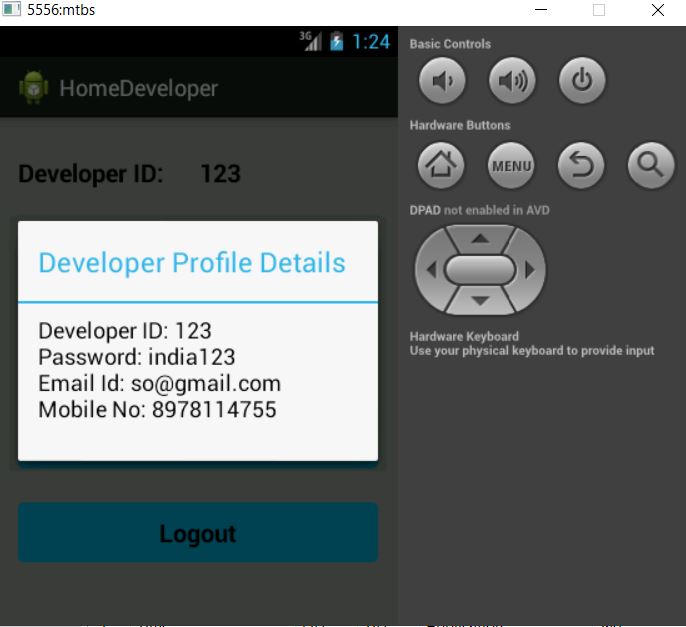
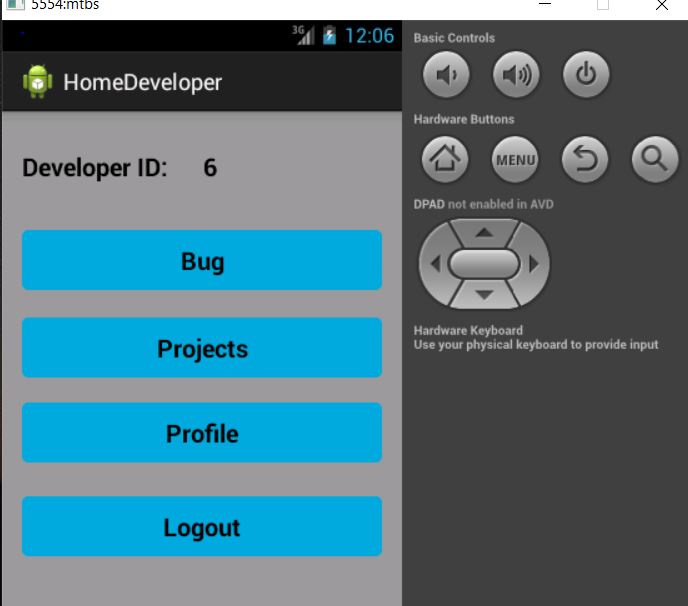
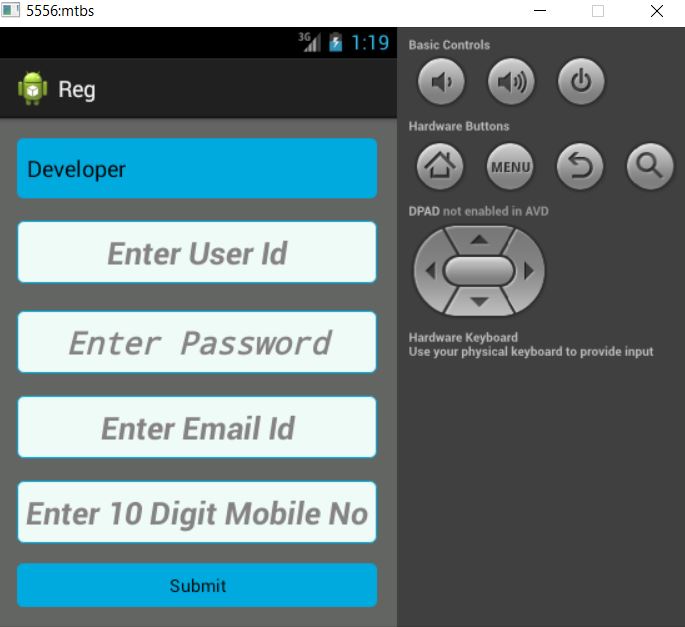
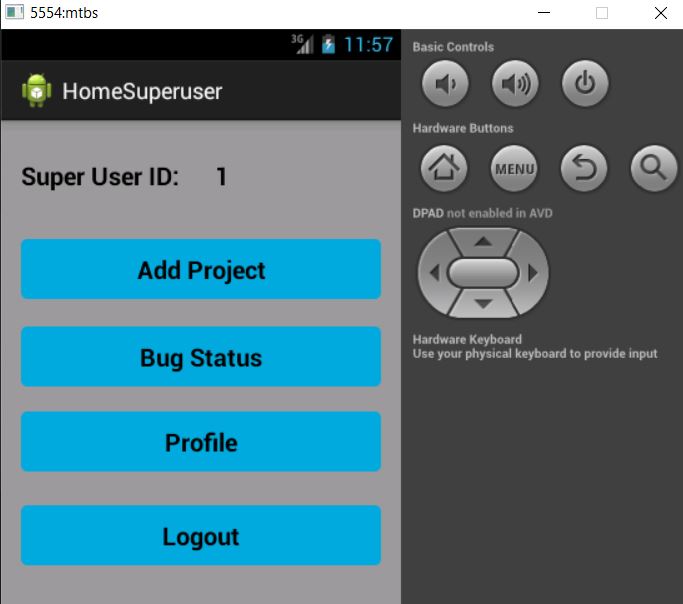
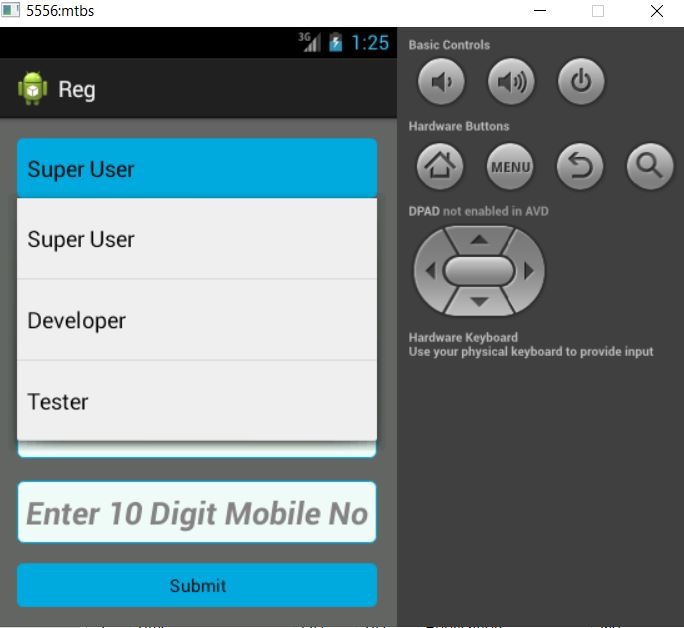
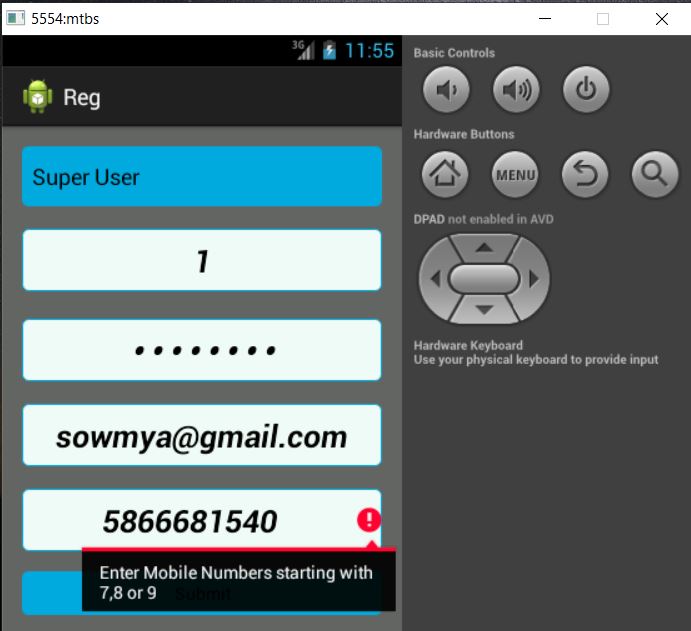
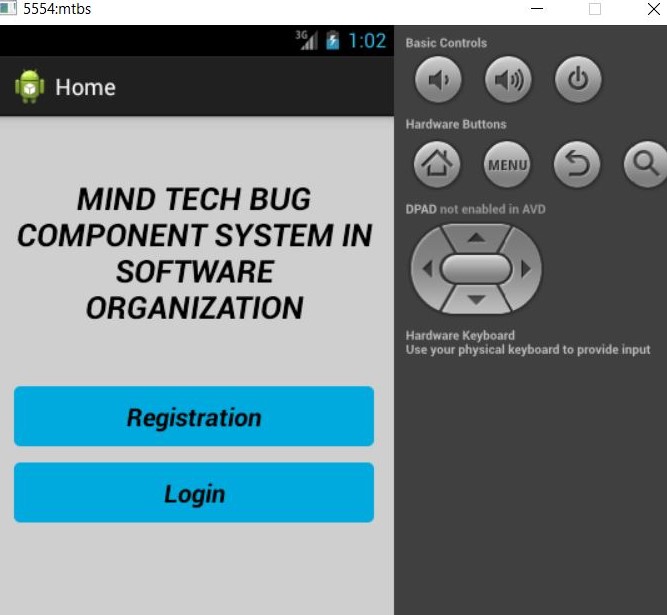
Module Name: Developer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO | TEST CASE | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | PASS/FAIL |
| 1 | Check Developer login functionality | Developer Username and Password | Login must be successful | Login success | Pass |
| 2 | Bugs | Check Bugs | Resolved to Bugs | Bugs  Resolved | Pass |

Module Name: Tester

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO | TEST CASE | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | PASS/FAIL |
| 1 | Check Tester Login Functionality | Tester Username and Password | Login must be successful | Login success | Pass |
| 2 | Check Bug Status | Open Bugs | Check bug details | Bugs  Resolved | Pass |

6 OUTPUT SCREENS



**7 Conclusion:**

At the end of this application it is a sophisticated approach for users to have a best selection and gives better performance for Administrator so that he can easily see bugs details occurred in the program and can be corrected. This makes the Tester to easy identify the bug and can be corrected for successful execution. This is convenient to Tester and super user to identify bug. There is also a facility for the Super user to check the status of the bug in the package.

8.FUTURE ENHANCEMENT:

The project has a very vast scope in the future.the project can be implemented on intranet in the future.project can be updated in near future as an when requirements of the same arises as it is very flexible in terms of expansion.with the proposed software of the datebase space manager ready and fully functional the client is now able to manage and hence run the entire work in a much better accurate and error free manner

## 9. **REFERENCES**

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

7th Edition, pp. 177-180.

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

**WEBSITES (or) REFERRED URLS:**

[www.android.com](http://www.android.com)

<http://developer.android.com/index.html>

[www.google.com](http://www.google.com)

<http://en.wikipedia.org/wiki/SQLite>

[www.abhiandroid.com](http://www.abhiandroid.com)

10. APPENDIX - A

## The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

|  |  |
| --- | --- |
| * Simple * Object oriented * Distributed * Multithreaded * Dynamic | * Architecture neutral * Portable * High performance * Robust * Secure |

Each of the preceding buzzwords is explained in [The Java Language Environment](http://www.oracle.com/technetwork/java/langenv-140151.html) , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine[1](https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html#FOOT) (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the [Java SE HotSpot at a Glance](http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html), perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.



Through the Java VM, the same application is capable of running on multiple platforms.

## The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. 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.

The Java platform has two components:

* The Java Virtual Machine
* The Java Application Programming Interface (API)

You've already been introduced to the Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, [What Can Java Technology Do?](https://docs.oracle.com/javase/tutorial/getStarted/intro/cando.html) highlights some of the functionality provided by the API.



The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

The terms"Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

# What Can Java Technology Do?

The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives you the following features:

* **Development Tools**: The development tools provide everything you'll need for compiling, running, monitoring, debugging, and documenting your applications. As a new developer, the main tools you'll be using are the javac compiler, the java launcher, and the javadoc documentation tool.
* **Application Programming Interface (API)**: The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more. The core API is very large; to get an overview of what it contains, consult the [Java Platform Standard Edition 8 Documentation](https://docs.oracle.com/javase/8/docs/index.html).
* **Deployment Technologies**: The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.
* **User Interface Toolkits**: The JavaFX, Swing, and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).
* **Integration Libraries**: Integration libraries such as the Java IDL API, JDBC API, Java Naming and Directory Interface (JNDI) API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program written in C++.
* **Write better code**: The Java programming language encourages good coding practices, and automatic garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans™ component architecture, and its wide-ranging, easily extendible API let you reuse existing, tested code and introduce fewer bugs.
* **Develop programs more quickly**: The Java programming language is simpler than C++, and as such, your development time could be up to twice as fast when writing in it. Your programs will also require fewer lines of code.
* **Avoid platform dependencies**: You can keep your program portable by avoiding the use of libraries written in other languages.
* **Write once, run anywhere**: Because applications written in the Java programming language are compiled into machine-independent bytecodes, they run consistently on any Java platform.
* **Distribute software more easily**: With Java Web Start software, users will be able to launch your applications with a single click of the mouse. An automatic version check at startup ensures that users are always up to date with the latest version of your software. If an update is available, the Java Web Start software will automatically update their installation.
* **Java** is a set of [computer software](https://en.wikipedia.org/wiki/Computer_software) and specifications developed by [James Gosling](https://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems), which was later acquired by the [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), that provides a system for developing [application software](https://en.wikipedia.org/wiki/Application_software) and deploying it in a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) computing environment. Java is used in a wide variety of [computing platforms](https://en.wikipedia.org/wiki/Computing_platform) from [embedded devices](https://en.wikipedia.org/wiki/Embedded_device) and [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone) to [enterprise servers](https://en.wikipedia.org/wiki/Enterprise_server) and [supercomputers](https://en.wikipedia.org/wiki/Supercomputer). [Java applets](https://en.wikipedia.org/wiki/Java_applet), which are less common than standalone Java applications, were commonly run in secure, [sandboxed](https://en.wikipedia.org/wiki/Sandbox_(computer_security)) environments to provide many features of native applications through being embedded in [HTML](https://en.wikipedia.org/wiki/HTML) pages.
* Writing in the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)) is the primary way to produce code that will be deployed as [byte code](https://en.wikipedia.org/wiki/Java_byte_code) in a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM); byte code [compilers](https://en.wikipedia.org/wiki/Compiler) are also available for other languages, including [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)). In addition, several languages have been designed to run natively on the JVM, including [Clojure](https://en.wikipedia.org/wiki/Clojure), [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)), and [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)). [Java syntax](https://en.wikipedia.org/wiki/Java_syntax) borrows heavily from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but object-oriented features are modeled after [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk) and [Objective-C](https://en.wikipedia.org/wiki/Objective-C).[[12]](https://en.wikipedia.org/wiki/Java_(software_platform)#cite_note-12) Java eschews certain low-level constructs such as [pointers](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) and has a very simple memory model where objects are [allocated on the heap](https://en.wikipedia.org/wiki/Dynamic_memory_allocation) (while some implementations e.g. all currently supported by Oracle, may use [escape analysis](https://en.wikipedia.org/wiki/Escape_analysis) optimization to allocate on the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation) instead) and all variables of object types are [references](https://en.wikipedia.org/wiki/Reference_(computer_science)). Memory management is handled through integrated automatic [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) performed by the JVM.

11 APPENDIX-B

CODE:

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

public class Aasignbug extends Activity implements OnClickListener{

TextView bid;

SQLiteDatabase db;

Button vieww,sub;

EditText did;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_aasignbug);

bid=(TextView)findViewById(R.id.textView3);

Bundle b = getIntent().getExtras();

bid.setText(b.getCharSequence("Name"));

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

vieww.setOnClickListener(this);

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

sub.setOnClickListener(this);

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

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists bug(tid varchar, bid varchar, type varchar,did varchar,bug varchar,status varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

}

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

if(v==vieww)

{

Cursor c=db.rawQuery("SELECT \* FROM dreg", null);

if(c.getCount()==0)

{

Toast.makeText(getApplicationContext(), "No Records Found", Toast.LENGTH\_SHORT).show();

return;

}

StringBuffer buffer=new StringBuffer();

while(c.moveToNext())

{

buffer.append("Developer Id: "+c.getString(1)+"\n");

buffer.append("Email: "+c.getString(2)+"\n");

buffer.append("Mobile No: "+c.getString(3)+"\n\n");

}

showMessage("Developer Details", buffer.toString());

}

if(v==sub)

{

if(did.getText().toString().trim().length()==0)

{

Toast.makeText(getApplicationContext(), "Enter Developer Id", Toast.LENGTH\_SHORT).show();

return;

}

Cursor c=db.rawQuery("SELECT \* FROM dreg WHERE id='"+did.getText()+"'", null);

if(c.moveToFirst())

{

db.execSQL("UPDATE bug SET did='"+did.getText()+"' WHERE bid='"+bid.getText()+"'");

Toast.makeText(getApplicationContext(), "Bug Assigned Successfully", Toast.LENGTH\_SHORT).show();

clr();

return;

}

else

{

Toast.makeText(getApplicationContext(), "Invalid Developer Id", Toast.LENGTH\_SHORT).show();

return;

}

}

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

public void clr()

{

did.setText("");

}

}

import java.util.ArrayList;

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

public class Aviewbugs extends Activity {

ImageView out;

TextView aaa;

SQLiteDatabase db;

ListView l;

EditText t1;

ArrayList<String> list1;

ArrayAdapter adapter;

String tid,bid,ssid,uemail,ty;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_aviewbugs);

final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

aaa=(TextView)findViewById(R.id.textView2);

aaa.setText(globalvariabel.GetUsername().toString());

out=(ImageView)findViewById(R.id.imageView1);

t1=(EditText)findViewById(R.id.search);

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists bug(tid varchar, bid varchar, type varchar,did varchar,bug varchar,status varchar);");

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

final ArrayList<String> list = new ArrayList<String>();

list1 = new ArrayList<String>();

Cursor res=db.rawQuery("SELECT \* FROM bug", null);

if(res.getCount()!=0)

{

while (res.moveToNext())

{

list.add("Tester ID: "+res.getString(0)+"\nBug ID: "+res.getString(1)+"\n"+"Type: "+res.getString(2)+"\n"+"Developer Id: "+res.getString(3)+"\n"+"Bug: "+res.getString(4)+"\n"+"Status: "+res.getString(5));

list1.add(res.getString(1));

ty=

tid=res.getString(0);

bid=res.getString(1);

}

}

adapter=new ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,list);

l.setAdapter(adapter);

// (close)

/// search (start)

t1.addTextChangedListener(new TextWatcher(){

@Override

public void afterTextChanged(Editable s) {

// TODO Auto-generated method stub

}

@Override

public void beforeTextChanged(CharSequence s, int start, int count,

int after) {

// TODO Auto-generated method stub

}

@Override

public void onTextChanged(CharSequence s, int start, int before,

int count) {

// TODO Auto-generated method stub

Aviewbugs.this.adapter.getFilter().filter(s);

}

});

l.setAdapter(adapter);

l.setOnItemClickListener(new OnItemClickListener()

{

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

String itemValue = (String) l.getItemAtPosition(position);

String bid = itemValue.substring(0, itemValue.indexOf('\n'));

int itemPosition = position;

String item=list1.get(itemPosition);

Intent i = new Intent(getApplicationContext(),Aasignbug.class);

Bundle b = new Bundle();

b.putString("Name", item);

i.putExtras(b);

startActivity(i);

}

});

}

//

///logout

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

}

import java.util.ArrayList;

import android.os.Bundle;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.AdapterView.OnItemClickListener;

public class Aviewreg extends Activity implements OnClickListener{

Spinner sp;

ImageView out;

TextView aaa;

SQLiteDatabase db;

ListView l;

EditText t1;

ArrayList<String> list1;

ArrayAdapter adapter;

Button sub;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_aviewreg);

sp=(Spinner)findViewById(R.id.spinner1);

ArrayAdapter<CharSequence> ar=ArrayAdapter.createFromResource(this, R.array.week, android.R.layout.simple\_list\_item\_1);

ar.setDropDownViewResource(android.R.layout.simple\_dropdown\_item\_1line);

sp.setAdapter(ar);

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

sub.setOnClickListener(this);

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists sreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists treg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

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

}

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

String u = sp.getSelectedItem().toString();

String myUser = "Super User";

if((v==sub) && u.equals(myUser))

{

final ArrayList<String> list = new ArrayList<String>();

list1 = new ArrayList<String>();

Cursor res=db.rawQuery("SELECT \* FROM sreg", null);

if(res.getCount()!=0)

{

while (res.moveToNext())

{

list.add("Super User ID: "+res.getString(1)+"\n Password: "+res.getString(2)+"\n"+"Email ID: "+res.getString(3)+"\n"+"Mobile: "+res.getString(4));

list1.add(res.getString(1));

}

}

adapter=new ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,list);

l.setAdapter(adapter);

}

String u1 = sp.getSelectedItem().toString();

String myUser1 = "Developer";

if((v==sub) && u1.equals(myUser1))

{

final ArrayList<String> list = new ArrayList<String>();

list1 = new ArrayList<String>();

Cursor res=db.rawQuery("SELECT \* FROM dreg", null);

if(res.getCount()!=0)

{

while (res.moveToNext())

{

list.add("Developer ID: "+res.getString(1)+"\n Password: "+res.getString(2)+"\n"+"Email ID: "+res.getString(3)+"\n"+"Mobile: "+res.getString(4));

list1.add(res.getString(1));

}

}

adapter=new ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,list);

l.setAdapter(adapter);

}

String u2 = sp.getSelectedItem().toString();

String myUser2 = "Tester";

if((v==sub) && u2.equals(myUser2))

{

final ArrayList<String> list = new ArrayList<String>();

list1 = new ArrayList<String>();

Cursor res=db.rawQuery("SELECT \* FROM treg", null);

if(res.getCount()!=0)

{

while (res.moveToNext())

{

list.add("Tester ID: "+res.getString(1)+"\n Password: "+res.getString(2)+"\n"+"Email ID: "+res.getString(3)+"\n"+"Mobile: "+res.getString(4));

list1.add(res.getString(1));

}

}

adapter=new ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,list);

l.setAdapter(adapter);

}

}

}

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

public class Dasignbug extends Activity implements OnClickListener{

TextView bid;

SQLiteDatabase db;

Button vieww,sub;

EditText did;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_dasignbug);

bid=(TextView)findViewById(R.id.textView3);

Bundle b = getIntent().getExtras();

bid.setText(b.getCharSequence("Name"));

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

sub.setOnClickListener(this);

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

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists bug(tid varchar, bid varchar, type varchar,did varchar,bug varchar,status varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

}

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

if(v==sub)

{

if(did.getText().toString().trim().length()==0)

{

Toast.makeText(getApplicationContext(), "Enter Developer Id", Toast.LENGTH\_SHORT).show();

return;

}

Cursor c=db.rawQuery("SELECT \* FROM bug WHERE bid='"+bid.getText()+"'", null);

if(c.moveToFirst())

{

db.execSQL("UPDATE bug SET status='"+did.getText()+"' WHERE bid='"+bid.getText()+"'");

Toast.makeText(getApplicationContext(), "Updated Successfully", Toast.LENGTH\_SHORT).show();

clr();

startActivity(new Intent(getApplicationContext(),Drepbug.class));

return;

}

else

{

Toast.makeText(getApplicationContext(), "Invalid Details", Toast.LENGTH\_SHORT).show();

return;

}

}

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

public void clr()

{

did.setText("");

}

}

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

public class Dprojectview extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_dprojectview);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.dprojectview, menu);

return true;

}

}

Button reg,log;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home);

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

reg.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(Home.this,Reg.class));

}

});

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

log.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(Home.this,Login.class));

}

});

}

@Override

public void onBackPressed() {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(), "you Cant Perform This Action", Toast.LENGTH\_LONG).show();

}

}

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

public class HomeAdmin extends Activity {

Button ab,bs,pr,log;

TextView aaa;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home\_admin);

final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

aaa=(TextView)findViewById(R.id.textView2);

aaa.setText(globalvariabel.GetUsername().toString());

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

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

pr=(Button)findViewById(R.id.button3);

log=(Button)findViewById(R.id.button4);

ab.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Aviewbugs.class));

}

});

bs.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Aviewreg.class));

}

});

log.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

});

pr.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

showMessage("DEVELOPED BY","MIND TECH BUG COMPONENT SYSTEM IN SOFTWARE ORGANIZATION \n GUIDE NAME: SURESH \n MOBILE NO:9700664440");

}

});

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

@Override

public void onBackPressed() {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(), "Please press Logout button goto Page", Toast.LENGTH\_LONG).show();

}

}

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

public class HomeDeveloper extends Activity {

Button ab,bs,pr,log;

TextView aaa;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home\_developer);

final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

aaa=(TextView)findViewById(R.id.textView2);

aaa.setText(globalvariabel.GetUsername().toString());

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

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

pr=(Button)findViewById(R.id.button3);

log=(Button)findViewById(R.id.button4);

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists pro(id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists sreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists treg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

ab.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Drepbug.class));

}

});

bs.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Cursor c=db.rawQuery("SELECT \* FROM pro", null);

if(c.getCount()==0)

{

showMessage("Error", "No records found");

return;

}

StringBuffer buffer=new StringBuffer();

while(c.moveToNext())

{

buffer.append("Project ID: "+c.getString(0)+"\n");

buffer.append("Name: "+c.getString(1)+"\n");

buffer.append("Company Name: "+c.getString(2)+"\n");

buffer.append("Description: "+c.getString(3)+"\n\n");

}

showMessage("Project Details", buffer.toString());

}

});

log.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

});

pr.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Cursor c=db.rawQuery("SELECT \* FROM dreg WHERE id='"+aaa.getText()+"'", null);

if(c.getCount()==0)

{

showMessage("Error", "No records found");

return;

}

StringBuffer buffer=new StringBuffer();

while(c.moveToNext())

{

buffer.append("Developer ID: "+c.getString(1)+"\n");

buffer.append("Password: "+c.getString(2)+"\n");

buffer.append("Email Id: "+c.getString(3)+"\n");

buffer.append("Mobile No: "+c.getString(4)+"\n");

}

showMessage("Developer Profile Details", buffer.toString());

}

});

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

@Override

public void onBackPressed() {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(), "Please press Logout button goto Page", Toast.LENGTH\_LONG).show();

}

}

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

public class HomeTester extends Activity {

Button ab,bs,pr,log;

TextView aaa;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home\_tester);

final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

aaa=(TextView)findViewById(R.id.textView2);

aaa.setText(globalvariabel.GetUsername().toString());

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

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

pr=(Button)findViewById(R.id.button3);

log=(Button)findViewById(R.id.button4);

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists sreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists treg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

ab.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Taddbug.class));

}

});

bs.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Tbugstatus.class));

}

});

log.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

});

pr.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Cursor c=db.rawQuery("SELECT \* FROM treg WHERE id='"+aaa.getText()+"'", null);

if(c.getCount()==0)

{

showMessage("Error", "No records found");

return;

}

StringBuffer buffer=new StringBuffer();

while(c.moveToNext())

{

buffer.append("Tester ID: "+c.getString(1)+"\n");

buffer.append("Password: "+c.getString(2)+"\n");

buffer.append("Email Id: "+c.getString(3)+"\n");

buffer.append("Mobile No: "+c.getString(4)+"\n");

}

showMessage("Tester Profile Details", buffer.toString());

}

});

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

@Override

public void onBackPressed() {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(), "Please press Logout button goto Page", Toast.LENGTH\_LONG).show();

}

}

import android.os.AsyncTask;

import android.os.Bundle;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class Login extends Activity {

EditText user, pass;

Button sub;

String u;

String p;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

setContentView(R.layout.activity\_login);

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

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

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

sub.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

if(user.getText().toString().equals("")||pass.getText().toString().equals("")){

Toast.makeText(Login.this, "Please enter all the fields..!", Toast.LENGTH\_LONG).show();

}else{

u = user.getText().toString();

p = pass.getText().toString();

db=openOrCreateDatabase("HAWK",SQLiteDatabase.CREATE\_IF\_NECESSARY,null);

}

if(user.getText().toString().equals("admin")&& pass.getText().toString().equals("admin")){

Toast.makeText(Login.this, "Welcome To Admin Home Page " + u , Toast.LENGTH\_LONG).show();

globalvariabel.Setusername(user.getText().toString().trim());

Intent a = new Intent(Login.this,HomeAdmin.class);

startActivity(a);

clear();

return;

}

Cursor cc = db.rawQuery("select \* from sreg where id= '"+u+"' and pass= '"+p+"' ", null);

if(cc.moveToFirst())

{String temp="";

if (cc != null) {

if(cc.getCount() > 0)

{

//return true;

scan g=new scan();

g.execute();

Toast.makeText(Login.this, "Welcome To SuperUser HomePage " + u , Toast.LENGTH\_LONG).show();

globalvariabel.Setusername(user.getText().toString().trim());

Intent b = new Intent(Login.this,HomeSuperuser.class);

startActivity(b);

clear();

return;

}else{

Toast.makeText(Login.this, "Login Failed...!", Toast.LENGTH\_LONG).show();

clear();

}

}

}

//lo

Cursor cc1 = db.rawQuery("select \* from dreg where id= '"+u+"' and pass= '"+p+"' ", null);

if(cc1.moveToFirst())

{String temp="";

if (cc1 != null) {

if(cc1.getCount() > 0)

{

//return true;

scan g=new scan();

g.execute();

Toast.makeText(Login.this, "Welcome To Developer Home Page " + u , Toast.LENGTH\_LONG).show();

globalvariabel.Setusername(user.getText().toString().trim());

Intent c = new Intent(Login.this,HomeDeveloper.class);

startActivity(c);

clear();

return;

}

else{

Toast.makeText(Login.this, "Login Failed...!", Toast.LENGTH\_LONG).show();

clear();

}

}

}

Cursor cc2 = db.rawQuery("select \* from treg where id= '"+u+"' and pass= '"+p+"' ", null);

if(cc2.moveToFirst())

{String temp="";

if (cc2 != null) {

if(cc2.getCount() > 0)

{

//return true;

scan g=new scan();

g.execute();

Toast.makeText(Login.this, "Welcome To Tester Home Page " + u , Toast.LENGTH\_LONG).show();

Intent c = new Intent(Login.this,HomeTester.class);

startActivity(c);

clear();

return;

}else{

Toast.makeText(Login.this, "Login Failed...!", Toast.LENGTH\_LONG).show();

clear();

}

}

}

else{

Toast.makeText(Login.this, "Login Failed...!", Toast.LENGTH\_LONG).show();

clear();

}

}//

});

}

public class scan extends AsyncTask<String, String, String>{

private ProgressDialog pd;

protected void onPreExecute() {

super.onPreExecute();

pd = new ProgressDialog(Login.this);

pd.setTitle("Please Wait!!");

pd.setMessage("Logging you In....");

pd.setMax(10);

pd.show();

}

@Override

protected String doInBackground(String... params) {

// TODO Auto-generated method stub

return null;

}

}

public void clear()

{

user.setText("");

pass.setText("");

}

}

import java.util.Timer;

import java.util.TimerTask;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.view.Menu;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

new Timer().schedule(new TimerTask() {

@Override

public void run() {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

},5000);

}

}

import java.util.Timer;

import java.util.TimerTask;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.view.Menu;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

new Timer().schedule(new TimerTask() {

@Override

public void run() {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

},5000);

}

}

import java.util.Timer;

import java.util.TimerTask;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.view.Menu;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

new Timer().schedule(new TimerTask() {

@Override

public void run() {

// TODO Auto-generated method stub

startActivity(new Intent(getApplicationContext(),Home.class));

}

},5000);

}

}

import android.os.Bundle;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.Toast;

public class Reg extends Activity implements OnClickListener{

Spinner sp;

String u,u1,u2;

EditText id,ps,em,mb;

Button sub;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_reg);

sp=(Spinner)findViewById(R.id.spinner1);

ArrayAdapter<CharSequence> ar=ArrayAdapter.createFromResource(this, R.array.week, android.R.layout.simple\_list\_item\_1);

ar.setDropDownViewResource(android.R.layout.simple\_dropdown\_item\_1line);

sp.setAdapter(ar);

db=openOrCreateDatabase("HAWK", Context.MODE\_PRIVATE, null);

db.execSQL("create table if not exists sreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists dreg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

db.execSQL("create table if not exists treg(emp varchar, id varchar, pass varchar,email varchar,mbl varchar);");

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

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

em=(EditText)findViewById(R.id.editText3);

mb=(EditText)findViewById(R.id.editText4);

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

sub.setOnClickListener(this);

}

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

String u = sp.getSelectedItem().toString();

String myUser = "Super User";

if((v==sub) && u.equals(myUser))

{

if(id.getText().toString().trim().length()==0||

ps.getText().toString().trim().length()==0||

em.getText().toString().trim().length()==0||mb.getText().toString().trim().length()==0)

{

Toast.makeText(getApplicationContext(),"Please Enter All Values", Toast.LENGTH\_SHORT).show();

return;

}

else if(!((ps.length()>=6)&&(ps.length()<=10)))

{

Toast.makeText(getApplicationContext(), "Enter min 6 and max 10 charecters",Toast.LENGTH\_SHORT).show();

return;

}

else if(mb.getText().toString().trim().length()!=10)

{

Toast.makeText(getApplicationContext(),"Enter 10 Digit Mobile No", Toast.LENGTH\_SHORT).show();

return;

}

else if(!android.util.Patterns.EMAIL\_ADDRESS.matcher(em.getText().toString()).matches())

{

Toast.makeText(getApplicationContext(),"Enter Proper Email ID", Toast.LENGTH\_SHORT).show();

return;

}

else if(mb.getText().length()==10){

if(mb.getText().toString().startsWith("7")||

mb.getText().toString().startsWith("8")||

mb.getText().toString().startsWith("9")){

}

else{

mb.setError("Enter Mobile Numbers starting with 7,8 or 9");

return;

}

}

Cursor c=db.rawQuery("SELECT \* FROM sreg WHERE email='"+em.getText()+"'", null);

if(c.moveToFirst())

{

Toast.makeText(getApplicationContext(),"Email ID Already Exists", Toast.LENGTH\_SHORT).show();

return;

}

db.execSQL("INSERT INTO sreg VALUES('"+sp.getSelectedItem()+"','"+id.getText()+

"','"+ps.getText()+"','"+em.getText()+"','"+mb.getText()+"');");

Toast.makeText(getApplicationContext(),"Super user Registration Successfully Done", Toast.LENGTH\_SHORT).show();

clearText();

return;

}

String u1 = sp.getSelectedItem().toString();

String myUser1 = "Developer";

if((v==sub) && u1.equals(myUser1))

{

if(id.getText().toString().trim().length()==0||

ps.getText().toString().trim().length()==0||

em.getText().toString().trim().length()==0||mb.getText().toString().trim().length()==0)

{

Toast.makeText(getApplicationContext(),"Please Enter All Values", Toast.LENGTH\_SHORT).show();

return;

}

else if(!((ps.length()>=6)&&(ps.length()<=10)))

{

Toast.makeText(getApplicationContext(), "Enter min 6 and max 10 charecters",Toast.LENGTH\_SHORT).show();

return;

}

else if(mb.getText().toString().trim().length()!=10)

{

Toast.makeText(getApplicationContext(),"Enter 10 Digit Mobile No", Toast.LENGTH\_SHORT).show();

return;

}

else if(!android.util.Patterns.EMAIL\_ADDRESS.matcher(em.getText().toString()).matches())

{

Toast.makeText(getApplicationContext(),"Enter Proper Email ID", Toast.LENGTH\_SHORT).show();

return;

}

else if(mb.getText().length()==10){

if(mb.getText().toString().startsWith("7")||

mb.getText().toString().startsWith("8")||

mb.getText().toString().startsWith("9")){

}

else{

mb.setError("Enter Mobile Numbers starting with 7,8 or 9");

return;

}