UNIVERSITY PROBLEMS DIARY

PROJECT REPORT

Project submitted in partial fulfillment of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SHAMEER BASHA N (R082477)

MOHAMMAD SHAFI N (R082487)

Under the supervision of

Mr.N.Satyanandaram (MSIT)

Dept. of Computer Science and Engineering,

RGUKT RKValley Campus

YSR dist – 516329.

April 2014

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is certify that the project report entitled **"University Problem Diary"** submitted by **Mohammad Shafi**, Roll No: **R082487** and **Shameer Basha**,Roll No:**R082477** to the Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies , RKValley, Kadapa, during the academic year 2013-2014 is a partial fulfillment for the award of Under graduate degree of **Bachelor of Technology** in **Computer Science and Engineering,** is a bonafide record carried out by him under my supervision. The project has fulfilled all the requirements as per as regulations of this institute and in my opinion reached the standard for submission.

S.Chandra Sekhar	N.Satyanandaram,
Branch Coordintor	Lecturer
Dept of CSE,	Dept. of C.S.E.,
RGUKT RKValley,	RGUKT RKValley,
Kadapa – 516329	Kadapa – 516329.

Date:
Place: RKValley

ACKNOWLEDGEMENT

I am highly indebted to **Mr.N.Satyanandaram** sir for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their kind co-operation, encouragementand their support in completing the project.

.

I would like to express my special gratitude and thanks to branch coordinator **Mr.Chandrasekhar**sir for giving me such attention and time.

I have taken efforts in this project. However, it could not have been possible without the kind support and help of many individuals and RGUKT. We would like to extend my sincere thanks to all of them.

My thanks and appreciation also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

ABSTRACT

Introducing the Android platform and the features of Android application, giving a detailed description of Android application framework from the prospective of developers. Here we are going to describe how the Android application development is based on the application framework and how it manages the notifications, memory, activities, services and processes without affecting the other applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and other application may make use of those capabilities (subject to security constraints enforced by the framework).This same mechanism allows components to be replaced by the user. As a young operating system, on one hand, Android could benefit from mature technology of other operating system. On the other hand, Android could also improve the blemish that appears in other operating systems. The permission that an application possessed has been defined clearly. In android applications, all the components would be independently launched. By using this services and security concepts we are Planned to develop app regarding day to day problems in Universities, Schools, and Offices etc.,

Here in this app the problems about academics, library, exams, placements etc could be posted by students. Those problems shall be immediately viewed by management and they will respond .In this app data could interact between students and management.

We tried to explore other ways to develop apps to run even more effectively. This will help the other developers where to concentrate for developing android application.

FEATURES

- Login and Registration for students as well as for faculty
- Updating the problems regarding Placements, Exams, Library, Mess, Academics, Hostel etc..,
- Retrieving all type of functionalities by using SQLite database

CONTENTS

Contents

1. INTRODUCTION:

This is anapp regarding day to day problems in Universities, Schools, and Offices etc. In this app problems about academics, library, exams, placement etc., could be posted by students. Those problems would be immediately viewed by management and they will respond. In this app data could be synchronized between students and management. This project explains about implementing an app for android mobiles which will help management to know about problems in different aspects and provides easy communication between higher authorities and students and provides immediate solution to the students.

In present trend usage of apps had become a new trend because of availability of web services on mobiles. By considering these improvements in mobile technology knowing information within university through mobile in less time can be useful for users. In this application initially users need to install app and update details like listing out different problems.

1.1 Android:

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

Android was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android was acquired by Google in 2005. After original release there have been number of updates in the original version of Android. Present version of android is 4.4.1(Kitkat).

1.2 MOTIVATION:

The main motivation of this application is to learn the mobile application development. We are always curious to know how things work on mobile. To provide a university problems diary application on Smartphone for the students with the facility to have synchronized data between the Smartphone and the student-management database.

2 SYSTEM ANALYSIS:

Analysis is the detailed study of the various operations performed by a system and their relationships within and outside of the system. A key question is: What must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether or not candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and problems handled by the present system.

2.1 Existing System:

- Time consuming process.
- Waiting time to meet authorities is more
- If we want any information, then it takes lot of time
- Problems may not be solved immediately

2.2 Proposed System Advantages:

- Updating an immediate problem
- Immediate response from higher authorities
- List of problems related to University
- List of recently updated problem
- Search problems by date and time

3 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the android version had to be chosen so that it is compatible with most of the Android devices. Hence Android 2.1 Éclair version was chosen.

3.1 Requirement Specification

3.1.1 Functional Requirements

- Graphical User interface which the user.
- SQLite that stores the information to be displayed to the user.

3.1.2 Software Requirements

For developing the application the following are the Software Requirements:

- Operating System: Windows 7
- Language: Android SDK, Java
- Database: SQLite
- Tools: Eclipse IDE, Android Plug-in for Eclipse
- Technologies used: Java, SQLite, XML and Android.
- Debugger: Android Dalvik Debug Monitor service

For running the application the following are the Software Requirements:

- Operating System: Android 2.3 or higher versions
- Network: Wi-Fi Internet or cellular Network

3.1.3 Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

For running the application:

- Device: Android version 2.1 and higher
- Minimum space to execute: 1.0MB

4 SYSTEM DESIGN:

4.1 Use Case Diagram:

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

User Case Diagram

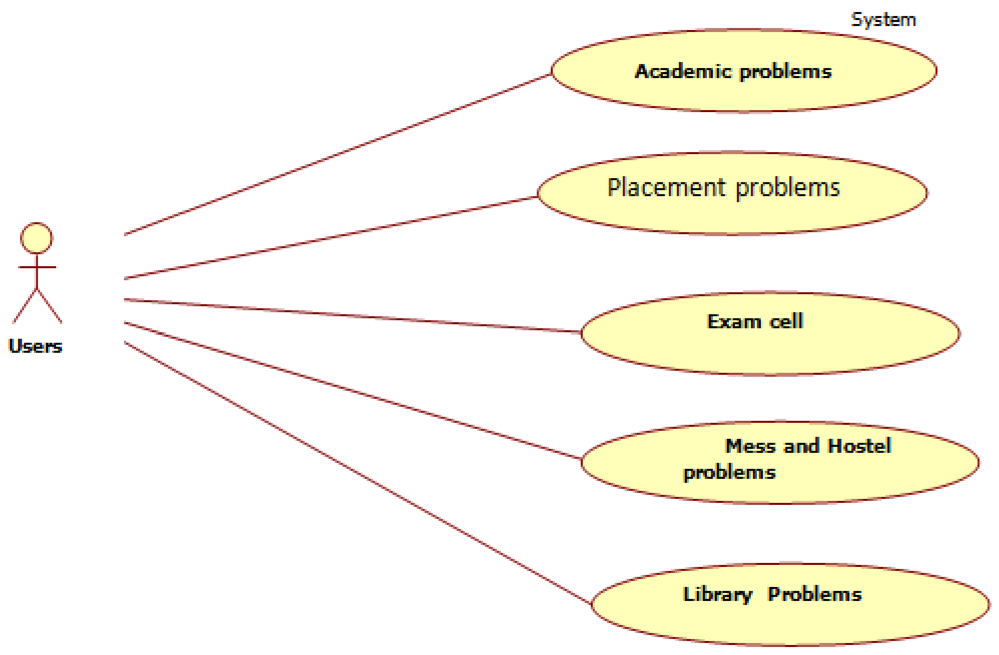


Figure 1:Data flow diagram

5 SOFTWARE ENVIRONMENT

5.1 Introduction to Android:

5.1.1 What is Android?

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

5.1.2 History of Android:

- In 2005 Google purchased Android Inc., November of 2007 - Google, under the Open Handset Alliance, announce Android. The first product to be released under the alliance is the mobile device operating system, Android in 2008.

- Google made available development tools and tutorials to aid would-be developers onto the new system.
- Help files, the platform software development kit (SDK) and Android, as a system, is a Java-based operating system that runs on the Linux 2.6 kernel. The system is very lightweight and full featured.

5.1.3 FEATURES:

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source Web Kit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQ Lite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and Wife** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE.

5.1.4 ANDROID ARCHITECTURE:

The following diagram shows the major components of the Android operating system. Each section is described in more detail below.

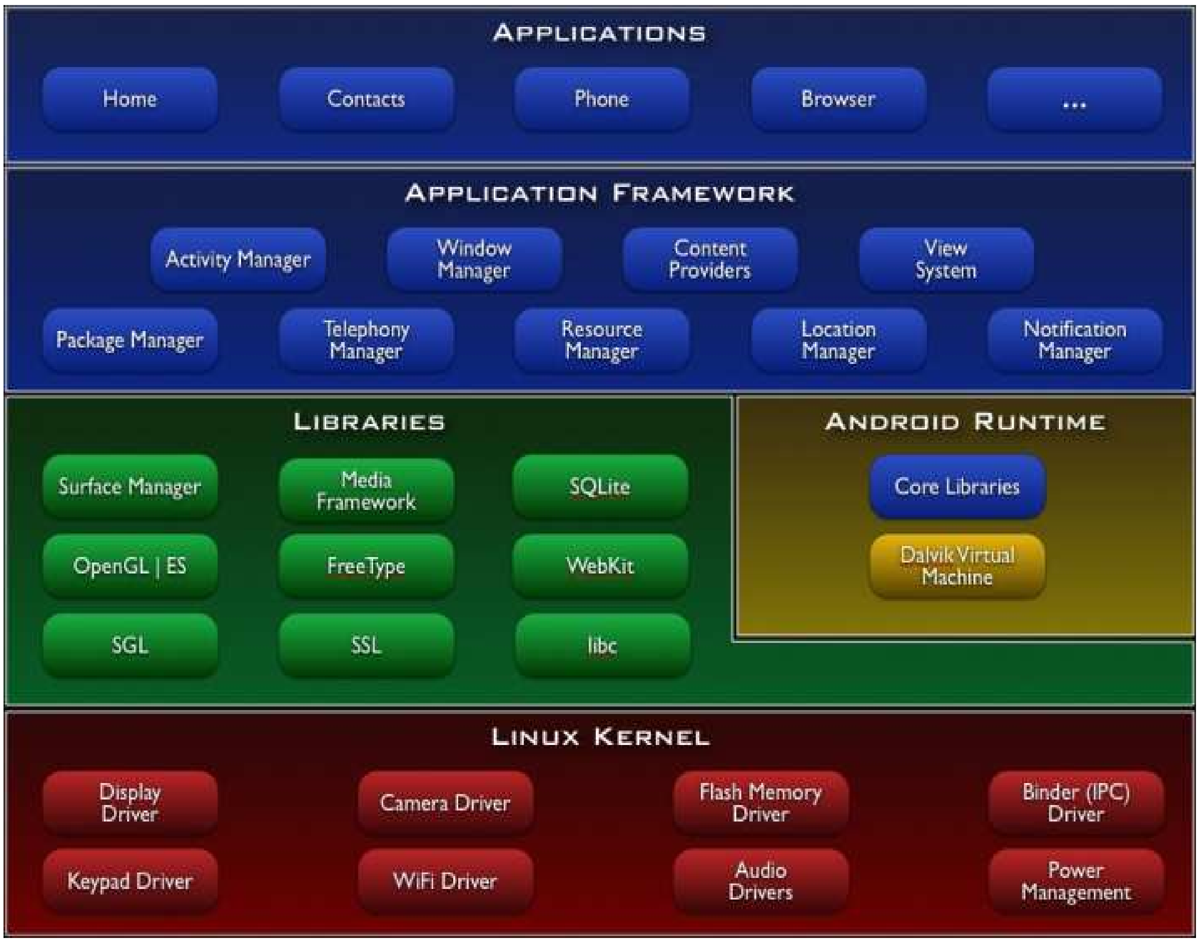


Figure 2:Android Architect

5.1.5 APPLICATIONS

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

5.2 ANDROID RUNTIME:

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the

DalvikExecutable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

5.2.1 LINUX KERNEL:

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of software stack.

5.3 Security and permissions:

5.3.1 Security concept in Android

During deployment on an Android device, the Android system will create a unique user and group ID for every Android application. Each application file is private to this generated user, e.g. other applications cannot access these files.

In addition each Android application will be started in its own process.

Therefore by means of the underlying Linux operating system, every Android application is isolated from other running applications.

If data should be shared, the application must do this explicitly, e.g. via a *service* or a *ContentProvider*.

5.3.2 Permission concept in Android

Android also contains a permission system. Android predefines permissions for certain tasks but every application can define additional permissions. An Android application declaresits required permissions in its *AndroidManifest.xml* configuration file. For example an application may declare that it requires access to the Internet.

Permissions have different levels. Some permissions are automatically granted by the Android system, some are automatically rejected.

In most cases the requested permissions will be presented to the user before installation of the application. The user needs to decide if these permissions are given to the application.

If the user denies permissions required by the application, this application cannot be installed. The check of the permission is only performed during installation and permissions cannot be denied or granted after the installation.

Not all users pay attention to the required permissions during installation. But some users do and they write negative reviews on Google Play.

5.4 Android applications and tasks

5.4.1 Application

An Android application consists of different Android components and additional resources. The Android systemknows *activities*, *services*, *broadcast receiver* and *content provider* as components.

5.4.2 Tasks across application borders

Android application components can connect to components of other Android applications to create *tasks*. For example an application which allows you to make a photo can start an email application and instruct this application to create a new email and attach a photo to this email.

5.5 Android user interface components

The following description gives an overview of the most important user interface related component and parts of an Android application.

5.5.1 Activity

An *activity* represents the visual representation of an Android application. *Activities* use *views,* i.e. user interface widgets as for example buttons and *fragments* to create the user interface and to interact with the user.

An Android application can have several *activities*.

5.5.2 Fragments

Fragments are components which run in the context of an *Activity*. A *Fragment* encapsulates application code so that it is easier to reuse it and to support different sized devices.

Fragments are optional components which allow you to reuse user interface and non user interface components for different devices configurations.

5.5.3 Views and layout manager

Views are user interface widgets, e.g. buttons or text fields. The base class for all *views* is the android.view.Viewclass. *Views* have attributes which can be used to configure their appearance and behavior.

A *layout manager* is responsible for arranging other *views*. The base class for these layout managers is theandroid.view.ViewGroup class which extends the View class.

Layout managers can be nestled to create complex layouts. You should avoid nestling them to deeply too deeply as this has a negative impact on the performance.

5.5.4 Device configuration specific layouts

The user interface for *Activities* is typically defined via XML files (layout files). It is possible to define defined layout file for different device configuration, e.g. based on the available width of the actual device running the application.

Fragments are designed to support such a setup.

5.6 Other Android components

Android has several more components which can be used in your Android application.

5.6.1 Intents

Intents are asynchronous messages which allow the application to request functionality from other Android components, e.g. from *services* or *activities*.

An application can call a component directly (*explicit Intent*) or ask the Android system to evaluate registered components based on the *intent* data (*implicit intents*). For example the application could implement sharing of data via intent and all components which allow sharing of data would be available for the user to select. Applications register themselves to an *intent* via an *intent Filter*.

Intents allow an Android application to start and to interact with components from other Android applications.

5.6.2 Services

Services perform tasks without providing a user interface. They can communicate with other Android components and notify the user via the notification framework in Android.

5.6.3 ContentProvider

A *content provider* provides a structured interface to application data. Via a *content provider* your application can share data with other applications. Android contains SQLite database which is frequently used in conjunction with a *content provider*. The SQLite database would store the data, which would be accessed via the *content provider*.

5.6.4 Broadcast Receiver

Broadcast receivers can be registered to receive system messages and *intents*. A *broadcast receiver* gets notified by the Android system, if the specified event occurs.

For example you can register *broadcast receivers* for the event that the Android system completed the boot processor or for the event that the state of the phone changes, e.g. someone is calling.

5.6.5 (HomeScreen) Widgets

Widgets are interactive components which are primarily used on the Android homescreen. They typically display some kind of data and allow the user to perform actions via them. For example a *Widget* could display a short summary of new emails and if the user selects an email, it could start the email application with the selected email.

5.6.6 Live Wallpapers

Live Wallpapers allow you to create animated backgrounds for the Android home screen.

5.7 Android Development Tools

5.7.1 Android SDK

The *Android Software Development Kit* (SDK) contains the necessary tools to create, compile and package Android application. Most of these tools are command line based.

The Android SDK also provides an Android device emulator, so that Android applications can be tested without a real Android phone. You can create *Android virtual devices* (AVD) via the Android SDK, which run in this emulator.

The Android SDK contains the *Android debug bridge* (adb) tool which allow us to connect to a virtual or real Android device.

5.7.2 Android Development Tools

Google provides the *Android Development Tools* (ADT) to develop Android applications with Eclipse. ADT is a set of components (plug-ins) which extend the Eclipse IDE with Android development capabilities.

ADT contains all required functionalities to create, compile, debug and deploy Android applications from the Eclipse IDE. ADT also allow us to create and start AVDs.

The Android Development Tools (ADT) provides specialized editors for resources files, e.g. layout files. These editors allow to switch between the

XML representation of the file and a richer user interface via tabs on the bottom of the editor.

5.7.3 Dalvik Virtual Machine

The Android system uses a special virtual machine, i.e. the *Dalvik Virtual Machine* to run Java based applications. Dalvik uses an own bytecode format which is different from Java bytecode.

Therefore you cannot directly run Java class files on Android, they need to get converted in the Dalvikbytecode format.

5.7.4 How to develop Android Applications

Android applications are primarily written in the Java programming language. The Java source files are converted to Java class files by the Java compiler.

The Android SDK contains a tool called *dx* which converts Java class files into a *.dex* (Dalvik Executable) file. All class files of one application are placed in one compressed *.dex* file. During this conversion process redundant information in the class files are optimized in the .dex file. For example if the same String is found in different class files, the *.dex* file contains only once reference of this String.

These dex files are therefore much smaller in size than the corresponding class files.

The *.dex* file and the resources of an Android project, e.g. the images and XML files, are packed into an *.apk* (Android Package) file. The program *aapt* (Android Asset Packaging Tool) performs this packaging.

The resulting *.apk* file contains all necessary data to run the Android application and can be deployed to an Android device via the *adb* tool.

The Android Development Tools (ADT) performs these steps transparently to the user.

If you use the ADT tooling you press a button the whole Android application (*.apk* file) will be created and deployed.

5.7.5 Resource editors

The ADT allows the developer to define certain artifacts, e.g. Strings and layout files, in two ways: via a rich editor, and directly via XML. This is done via multi-page editors in Eclipse. In these editors you can switch between both representations by clicking on the tab on the lower part of the screen.

For example if you open the *res/layout/main.xml* file in the *Package Explorer* View of Eclipse, you can switch between the two representations as depicted in the following screenshot.

5.8 Android Application Architecture

5.8.1 AndroidManifest.xml

The components and settings of an Android application are described in the *AndroidManifest.xml* file. For example all*activities* and *services* of the application must be declared in this file.

It must also contain the required permissions for the application. For example if the application requires network access it must be specified here.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifestxmlns:android="http://schemas.android.com/apk/re
s/android"

package="de.vogella.android.temperature"

android:versionCode="1"

android:versionName="1.0">

<applicationandroid:icon="@drawable/icon"android:label="@string
/app_name">

<activityandroid:name=".Convert"

android:label="@string/app_name">

<intent-filter>

<actionandroid:name="android.intent.action.MAIN" />

<categoryandroid:name="android.intent.category.LAUNCHER"
/>

</intent-filter>

</activity>



</application>

<uses-sdkandroid:minSdkVersion="9" />



</manifest>
```

The *package* attribute defines the base package for the Java objects referred to in this file. If a Java object lies within a different package, it must be declared with the full qualified package name.

Google Play requires that every Android application uses its own unique package. Therefore it is a good habit to use your reverse domain name as package name. This will avoid collisions with other Android applications.

android:versionName and *android:versionCode* specify the version of your application. *versionName* is what the user sees and can be any String.

versionCode must be an integer. The Android Market determine based on the versionCode, if it should perform an update of the applications for the existing installations. You typically start with "1" and increase this value by one, if you roll-out a new version of your application.

The <*activity*> tag defines an *activity*, in this example pointing to
the Convert class in thede.vogella.android.temperature package. An intent
filter is registered for this class which defines that this*activity* is started once
the application starts (action *android:name="android.intent.action.MAIN"*).
The categorydefinition *category*
android:name="android.intent.category.LAUNCHER" defines that this
application is added to the application directory on the Android device.

The @string/app_name value refers to resource files which contain the actual value of the application name. The usage of resource file makes it easy to provide different resources, e.g. strings, colors, icons, for different devices and makes it easy to translate applications.

The *uses-sdk* part of the *AndroidManifest.xml* file defines the minimal SDK version for which your application is valid. This will prevent your application being installed on unsupported devices.

5.8.2 Activities and Lifecycle

The Android system controls the lifecycle of your application. At any time the Android system may stop or destroy your application, e.g. because of an incoming call. The Android system defines a lifecycle for *activities* via predefined methods. The most important methods are:

- onSaveInstanceState() - called after the Activity is stopped. Used to save data so that the Activitycan restore its states if re-started
- onPause() - always called if the Activity ends, can be used to release resource or save data
- onResume() - called if the Activity is re-started, can be used to initialize fields

5.9 INTRODUCTION TO JAVA

5.9.1 HISTORY OF JAVA:

Java language was developed by James Gosling and his team at sun micro systems and released formally in 1995. Its former name is oak. Java Development Kit 1.0 was released in 1996.

Overview of Java

Java is loosely based on C++ syntax, and is meant to be Object-Oriented Structure of java is midway between an interpreted and a compiled language .java programs are compiled by the java compiler into Byte Codes which are secure and portable across different platforms. These byte codes are essentially instructions encapsulated in single type, to what is known as ajava virtual machine (JVM) which resides instandard browser.

JVM verifies these byte codes when downloaded by the browser for integrity. JVM is available for almost all OS. JVM converts these byte codes into machine specific instructions at runtime.

5.9.2 FEATURES OF JAVA :

- Java is object-oriented language and supports encapsulation, inheritance, polymorphism and dynamic binding, but does not support multiple inheritance. Everything in java is an object except some primitive datatypes.
- Java is portable,distributed, robust, secured, high performing and dynamic in nature.
- Java supports multithreading. There for different parts of the program can be executed at the same time

5.9.3 JAVA AND INTERNET:

Java is strongly associated with internet and known as internet programming language. Internet users can use java to create applet programs and run them locally using java enabled browser search as hot java. Applets can be downloaded from remote machine via internet and run it on local machine.

5.9.4 JAVA AND WORLD WIDE WEB:

World Wide Web is an open ended information retrieval system designed to be used in the distributed environment. This system contains web pages that provide both information and controls. We can navigate to a new web page in any direction. This is made possible worth HTML java was meant to be used in distributed environment such as internet. So java could be easily incorporated into the web system and is capable of supporting animation graphics, games and other special effect. The web has become more dynamic and interactive with support of java. We can run a java program on remote machine over internet with the support of web.

5.9.5 JAVA ENVIRONMENT:

Java environment includes a large number of tools which are part of the system known as java development kit (JDK) and hundreds of classes, methods, and interfaces grouped into packages forms part of java standard library(JSL).

5.9.6 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 are then interpreted on each platform by the runtime environment .java also provides stringent compile and runtime checking and automatic memory management in order to ensure solid code.

5.9.7 JAVA VIRTUAL MACHINE:

When we compile the code, java compiler creates machine code (byte code) for a hypothetical machine called java virtual machine (JVM). The JVMwill execute the byte code and overcomes the issue of portability .the code is written and compile for one machine and interpreted all other machines. This machine is called java virtual machine.

5.9.8 Batch updates:

The batch update feature allows an application to submit multiple update statements(insert/update/delete) in a single request to the database. This can provide a dramatic increase in performance when a large number of update statements need to be executed.

5.9.9 Advanced data types:

Increased support for storing persistent Java programming language objects (Java objects)and a mapping for SQL99 data types such as binary large objects, and structuredtypes, have been added to the JDBC API. An application may also customize the map-ping of SQL99 structured types into Java programming language classes.

A Simple Application

Consider the following trivial application that prints "hi there" to standard

output:Java Basics

© 1996-2003 jGuru.com. All Rights Reserved. Java Basics -5

```
public class TrivialApplication {
  // args[0] is first argument
  // args[1] the second
public static void main(String args[]) {
System.out.println("hi there");
  }
}
```

The command java TrivialApplication tells the Java runtime system to beginwith the class file TrivialApplication.class and to look in that file for a method with the signature:

```
public static void main(String args[]);
```

The main() method will always reside in one of your class files. The Java

language does not allow methods outside of class definitions. The class, in effect, creates scoped symbol StartingClassName.main for your main() method.

Applet Execution

An applet is a Java program that runs within a Java-compatible WWW browser or in an appletviewer. To execute your applet, the browser: Creates an instance of your applet Sends messages to your applet to automatically invoke predefined lifecycle methods the predefined methods automatically invoked by the runtime system are:

init(). This method takes the place of the Applet constructor and is only calledonce during applet creation. Instance variables should be initialized in this method. GUI components such as buttons and scrollbars should be added to the GUI in this method.

start(). This method is called once after init() and whenever your applet is revisited by your browser, or when you deconify your browser. This methodshould be used to start animations and other threads.

paint(Graphics g): This method is called when the applet drawing area needs

to be redrawn. Anything not drawn by contained components must be drawn in this method. Bitmaps, for example, are drawn here, but buttons are not becausethey handle their own painting.

stop(): This method is called when you leave an applet or when you iconifyyour browser. The method should be used to suspend animations and otherJava Basics

destroy(). This method is called when an applet terminates, for example, whenquitting the browser. Final clean-up operations such as freeing up systemresources with dispose() should be done here. The dispose() method of Frameremoves the menu bar. Therefore, do not forget to call super.dispose() if youoverride the default behavior.

The basic structure of an applet that uses each of these predefined methods is:

importjava.applet.Applet;

// include all AWT class definitions

importjava.awt.*;

```
public class AppletTemplate extends Applet {
public void init() {
   // create GUI, initialize applet
  }
public void start() {
   // start threads, animations etc...
  }
public void paint(Graphics g) {
   // draw things in g
  }
public void stop() {
   // suspend threads, stop animations etc...
  }
public void destroy() {
   // free up system resources, stop threads
  }
}
```

All you have to do is fill in the appropriate methods to bring your applet to life. If you don't need to use one or more of these predefined methods, simply leave them out of your applet. The applet will ignore messages from the browser attempting to invoke any of these methods that you don't use.

6 IMPLEMENTATION

Debugging of the application throughout the development is done using Dalvik Debug Monitor Server (DDMS). DDMS provides port-forwarding services, screen capture on the device, thread and heap information on the device, *logcat*, process, and radio state information, incoming call and SMS spoofing, location data spoofing etc.[1]. DDMS is also used to verify the location based services implemented in the application.

6.1 Graphical User Interface

The user interface is kept simple and understandable. The user need not take any additional effort to understand the functionality and navigation in the application. The colors are chosen in such a way that user can easily understand where the input has to be given. Hints are given to help the user in giving the correct input

The following are the main screens and features in this application.

- SplashScreen
- HomeScreen
- List of Problems
- Every individual problem Details
- Menu Details
- Updating Problem Details
- Displaying problems Details

6.2 SplashScreen Activity:

This is the start page of the application only used for Splash Screen. The Screen displayed until the **HomeActivity**is loaded. The time to load the HomeActivity is calculated by using the **Handler** class and **Threads.**

6.2.1 HomeScreenActivity:

The home screen activity is contains the user login page. User enters the login and password.

6.2.2 List of Problems:

This class is main part the application, it will display the options for to write the different type of queries.

6.2.3 Every individual queries:

This activity explains the different type of problems. For placement there is separate column, for mess problems different columns will be assigned.

6.2.4 Updating Problem details:

This activity is explains about updating the problem details. If you want change the existing data. This can update the previous data.

6.2.5 Menu Details:

This is explains about the menu if we click on menu we can get the Displaying different type problems.

6.2.6 Displaying different type of problems:

This Activity is explains about name of particular type of function like library,hostel etc.,

MODULES DESCRIPTION:

1. **Academic problems :**academic problems will be sent to academic coordinator
2. **Library Problems :**All Library problems regarding books,submissions, taking et
3. **Placement cell, Exam cell**
4. **Mess and Hostel problems**
5. **Retrieving database:** Database subroutines will be sent to kinds of authorities

.

6.3 Sample Code:

6.3.1 Sample code for Login:

```
public class Login extends Activity{

        EditText ed,ed1;

        ImageButton login;
        String s1,s2;

        Button b1,b2;

        public void onCreate(Bundle savedInstanceState){


                super.onCreate(savedInstanceState);

                setContentView(R.layout.login);

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

          ed1=(EditText)findViewById(R.id.editText2);

        login=(ImageButton)findViewById(R.id.imageButton1);

        login.setOnClickListener(new OnClickListener(){

                        public void onClick(View arg0) {

                                s1=ed.getText().toString();

                                s2=ed.getText().toString();

                                if(s1.equals("sha") && s2.equals("sha")){

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

                                        startActivity(i);

                                        }

                                else

                                        Toast.makeText(getApplicationContext(), "Login Failed Try
again", 0).show();

                        }

          });

          b1=(Button)findViewById(R.id.button1);

        b1.setOnClickListener(new OnClickListener(){

                        public void onClick(View v) {

                                // TODO Auto-generated method stub

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

                                startActivity(i);

                        }

          });
```

```
      b2=(Button)findViewById(R.id.button2);

    b2.setOnClickListener(new OnClickListener(){

                public void onClick(View v) {

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

                        startActivity(i);

                }

      });

    }

}
```

6.3.2 List of queries(placement,exams etc.,):

```
publicclass Academics extends Activity {
      publicvoidonCreate(Bundle savedInstancestate){
            super.onCreate(savedInstancestate);
            setContentView(R.layout.academics);
      }

}
```

6.3.3 Database code:

```
importandroid.app.Activity;
importandroid.content.ContentValues;
importandroid.content.Intent;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.view.View.OnClickListener;
importandroid.widget.Button;
importandroid.widget.EditText;
importandroid.widget.Toast;

publicclass signup extends Activity {
      EditTextfirstname,lastname, rollno, password,repassword,email,mobile;
DBHelperdbhelper;
    String fn,ln,rn,pass,repass,em,mob;
    Button submit,clear;
      SQLiteDatabasedb;
publicvoidonCreate(Bundle savedInstanceState){

            super.onCreate(savedInstanceState);
            setContentView(R.layout.signup);
            firstname = (EditText) this.findViewById(R.id.firstname);
            lastname = (EditText) this.findViewById(R.id.lastname);
            rollno = (EditText) this.findViewById(R.id.rollno);
      password = (EditText) this.findViewById(R.id.password);
      repassword = (EditText) this.findViewById(R.id.repassword);
      email = (EditText) this.findViewById(R.id.email);
      mobile = (EditText) this.findViewById(R.id.mobile);
```

```
        fn=firstname.getText().toString();
        ln=lastname.getText().toString();
        rn=rollno.getText().toString();
        pass=password.getText().toString();
        repass=repassword.getText().toString();
        em=email.getText().toString();
        mob=mobile.getText().toString();

                dbhelper=newDBHelper(this);
                db=dbhelper.getWritableDatabase();
                submit=(Button)findViewById(R.id.submit);
                clear=(Button)findViewById(R.id.clear);
        start();
    }
protectedvoidonStart(){
        db.execSQL("create table if not exists signup ( "+Database.FIRSTNAME + "
TEXT," +Database.LASTNAME + " TEXT," +Database.PASSWORD  + " TEXT,"
+Database.ROLLNO + " TEXT," +Database.MOBILE + " TEXT," +Database.EMAIL + "
TEXT)");
        super.onStart();

    }
privatevoid start(){
        submit.setOnClickListener(newOnClickListener(){

        publicvoidonClick(View v) {
        ContentValues values = newContentValues();
        values.put(Database.FIRSTNAME, fn);
        values.put(Database.LASTNAME, ln);
        values.put(Database.ROLLNO, rn);
        values.put(Database.PASSWORD, pass);
        values.put(Database.EMAIL, em);
        values.put(Database.MOBILE, mob);
        long rows=db.insert("signup", null, values);
        db.close();
        if(rows>0)
        {
        Toast.makeText(getApplicationContext(), "Successfully Registered",
        Toast.LENGTH_LONG).show();
        Intent i=newIntent(getApplicationContext(),Login.class);
        startActivity(i);
        }
        else
        Toast.makeText(getApplicationContext(),"NotRegistered",Toast.LENGTH_LONG).s
        how();
        }
        });
```

6.4 DATABASE:

The location info latitude, longitude, address, category and other location information’s are stored in SQLite data storage.

Databases have been an integral part of software applications since the dawn of the commercial application market several decades ago. As crucial as database management systems are, they also come with a large footprint, and considerable overhead in system resources and administration complexity. As software applications become less monolithic and more modular, a new type of database can be a better fit than the larger and more complex traditional database management systems. Embeddable databases run directly in the application process, offer zero-configuration run modes, and have very small footprints. This article introduces the popular SQLite database engine and describes how to use it in application development.

6.4.1 SQLite:

SQLite is an open source embeddable database engine written in C by D. Richard Hipp. It is entirely self-contained with no external dependencies. It was introduced as an option in PHP V4.3 and is built into PHP V5. SQLite supports much of the SQL92 standard, runs on all major operating systems, and has support for the major computer languages. SQLite is also surprisingly robust. Its creator conservatively estimates that it can handle a Web site with a load of up to 100, 00 hits a day, and there have been cases where SQLite has handled a load 10 time.The complete SQLite locking semantics are documented at SQLite.org.

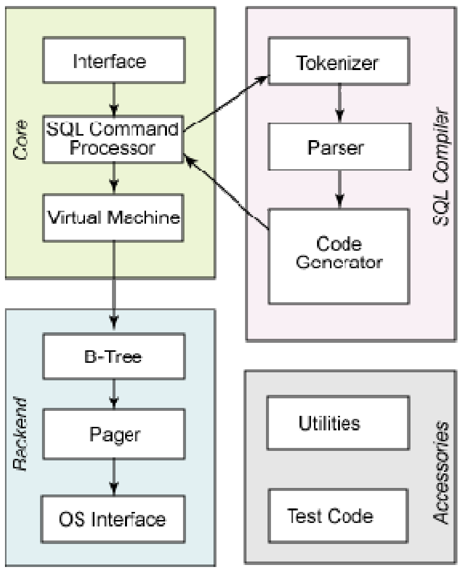


Figure 3:Internal architecture of SQLite

Features of SQLite:

- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Implements most of SQL92. (Features not supported)
- A complete database is stored in a single cross-platform disk file.
- Supports terabyte-sized databases and gigabyte-sized strings and blobs.
- Small code footprint: less than 400KiB fully configured or less than 250KiB with optional features omitted.
- Faster than popular client/server database engines for most common operations.
- Simple, easy to use API.
- Written in ANSI-C. TCL bindings included. Bindings for dozens of other languages available separately.
- Well-commented source code with 100% branch test coverage.
- Available as a single ANSI-C source-code file that you can easily drop into another project.
- Self-contained: no external dependencies.

- Cross-platform: UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT) are supported out of the box. Easy to port to other systems.
- Sources are in the public domain. Use for any purpose.
- Comes with a standalone command-line interface (CLI) client that can be used to administer SQLite databases.

7 SYSTEM TESTING

7.1 Unit Testing:

In unit testing, various modules have been tested individually. This has been done manually to test if the expected result is actually seen on the screen. The following are test cases with the help of which the application has been tested.

7.2 Performance Testing:

Performance testing has been done to measure the responsiveness of the application to the workload such as increasing users' requests. JMeter was used to create the users and to analyze the performance. The parameters were chosen randomly till the application performed consistently. The test has been done on a home Wi-Fi network with a speed of 5Mbps

7.3 Compatibility Testing:

This application was mainly designed for android phones as it helps the users to post problems to the officials. Generally they try to carry something handy like cell phones with them and not the tablets. Different android phones have different screen sizes and resolution. The application has been tested for its compatibility with different screen sizes on the emulator.

8 SAMPLE SCREENS

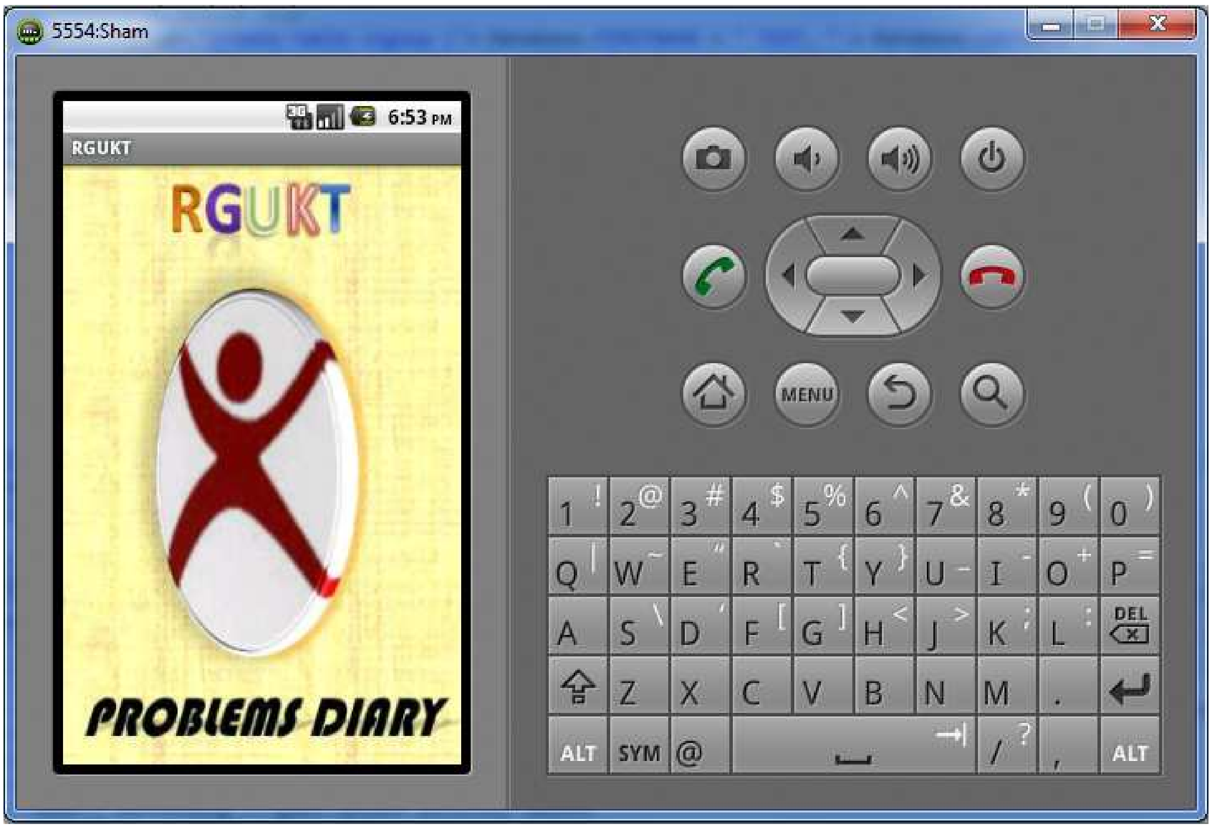


Figure 4:Splash screen

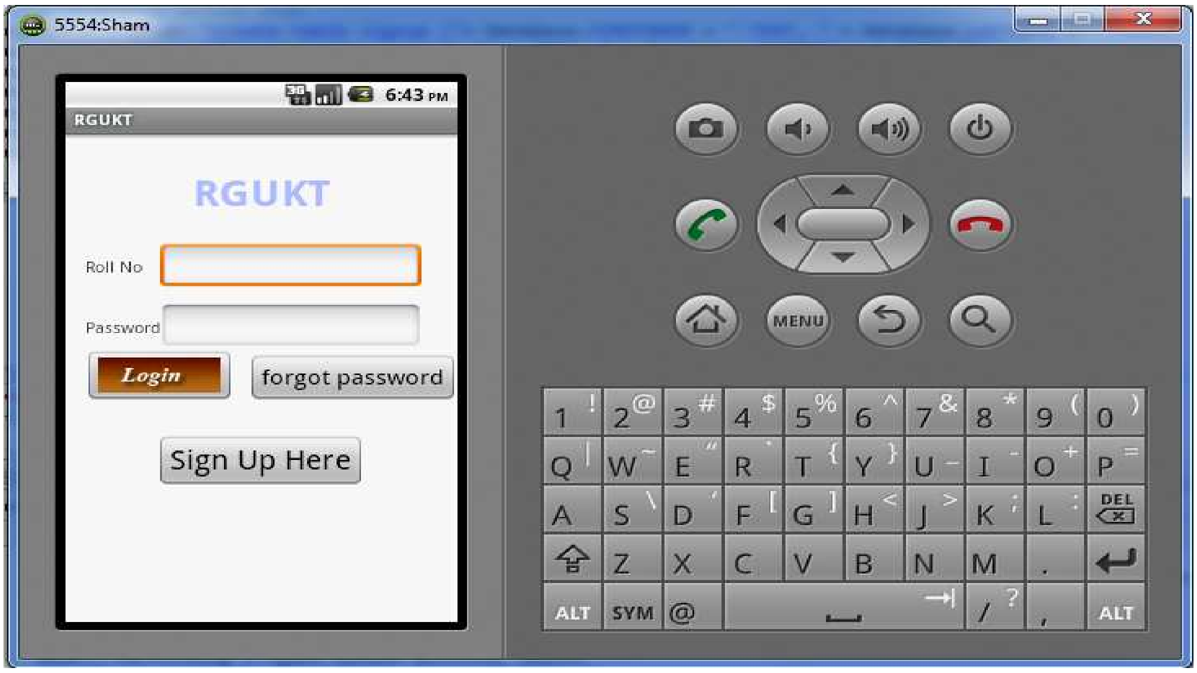
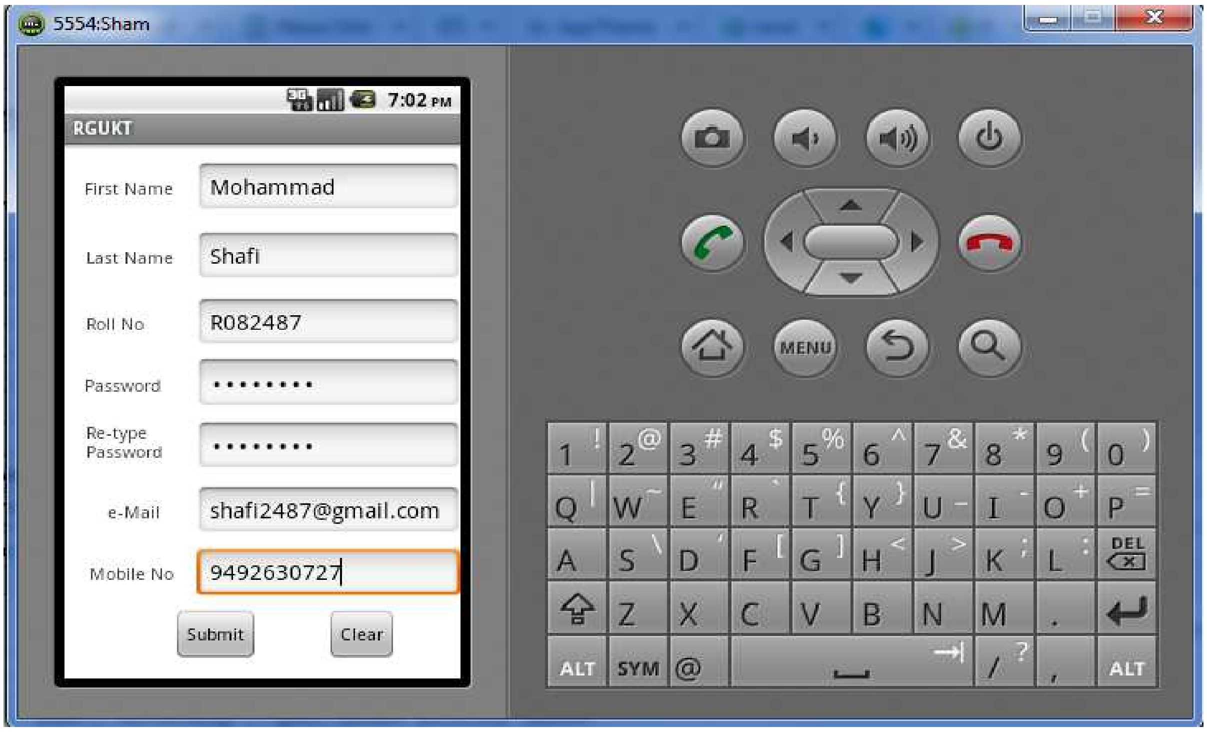


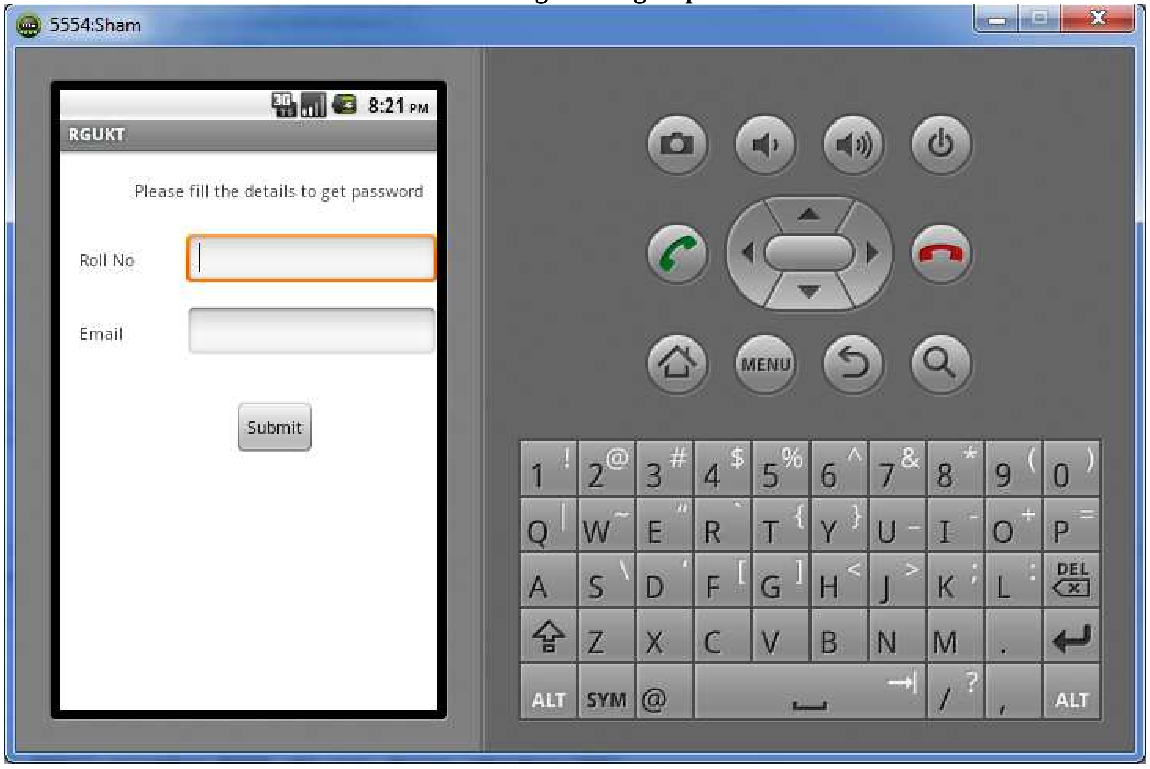
Figure 5:Home Screen


5554:Sham

RGUKT

First Name	Mohammad
Last Name	Shafi
Roll No	R082487
Password	•••••••••
Re-type Password	•••••••••
e-Mail	shafi2487@gmail.com
Mobile No	9492630727

Figure 6:sign-up


5554:Sham

RGUKT

Please fill the details to get password

Roll No	
Email	

Figure 7:Forgot password

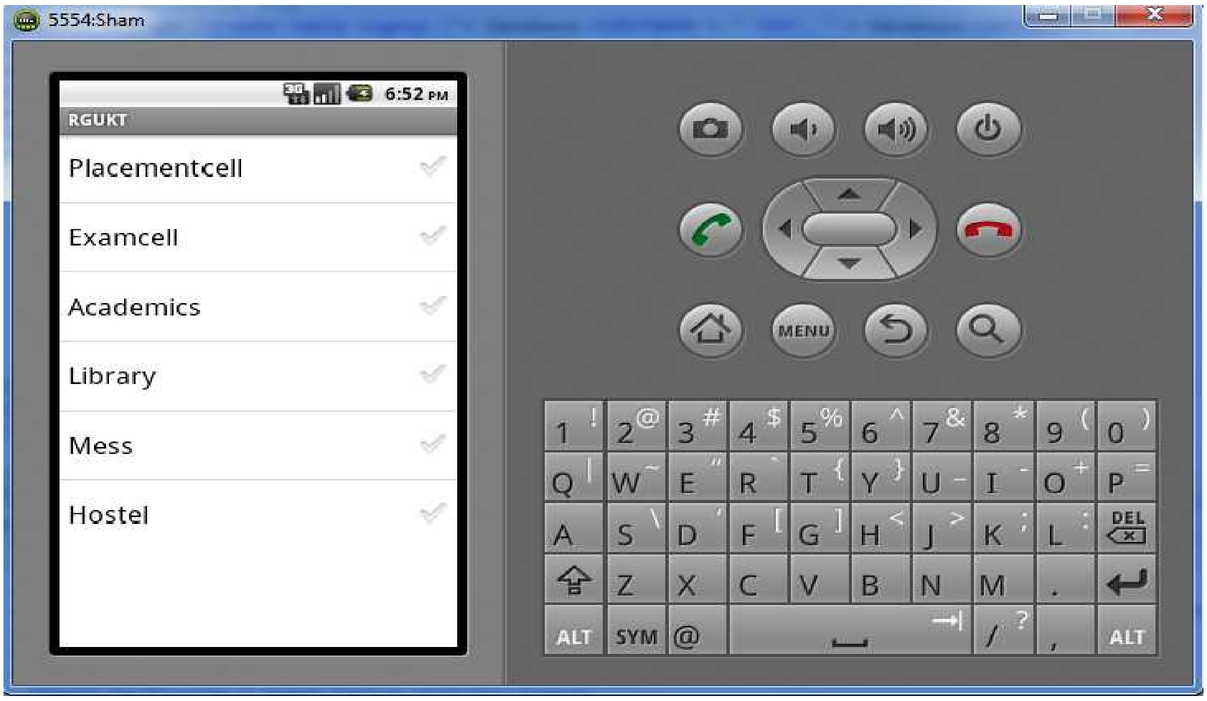
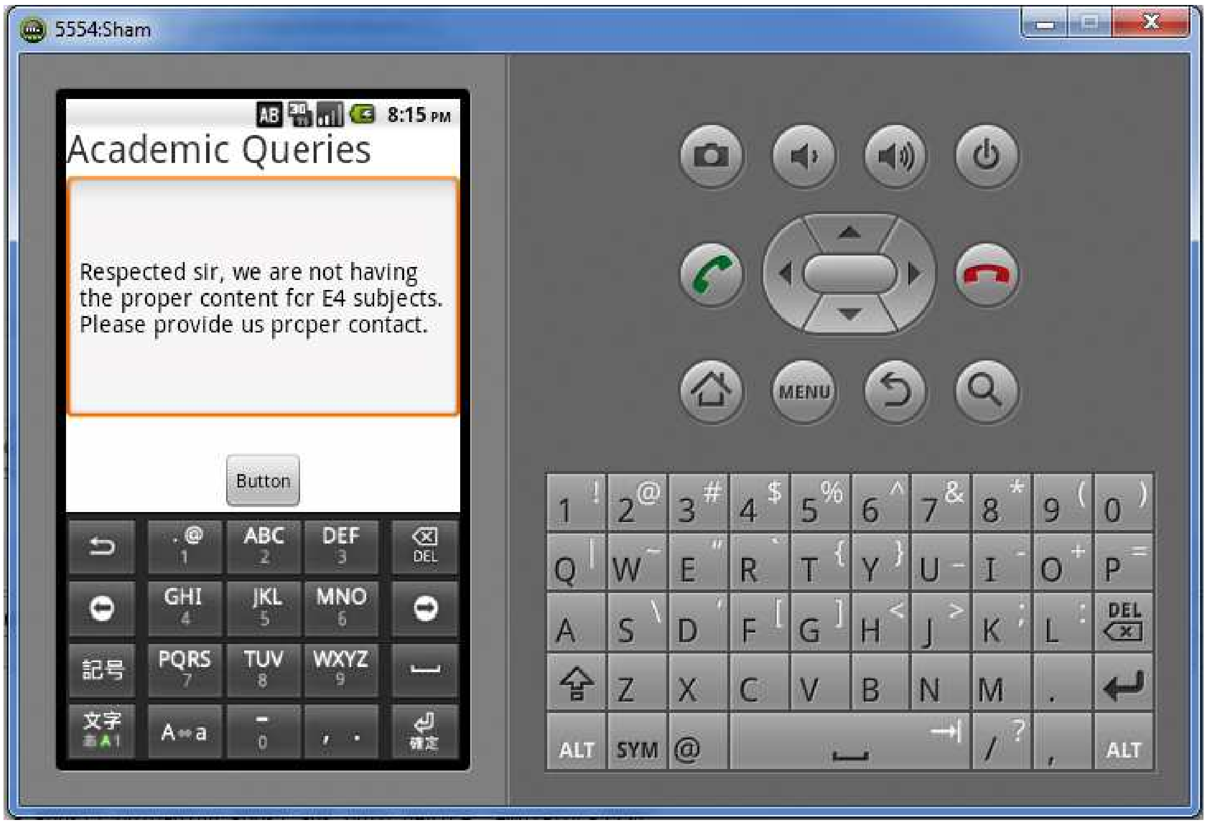
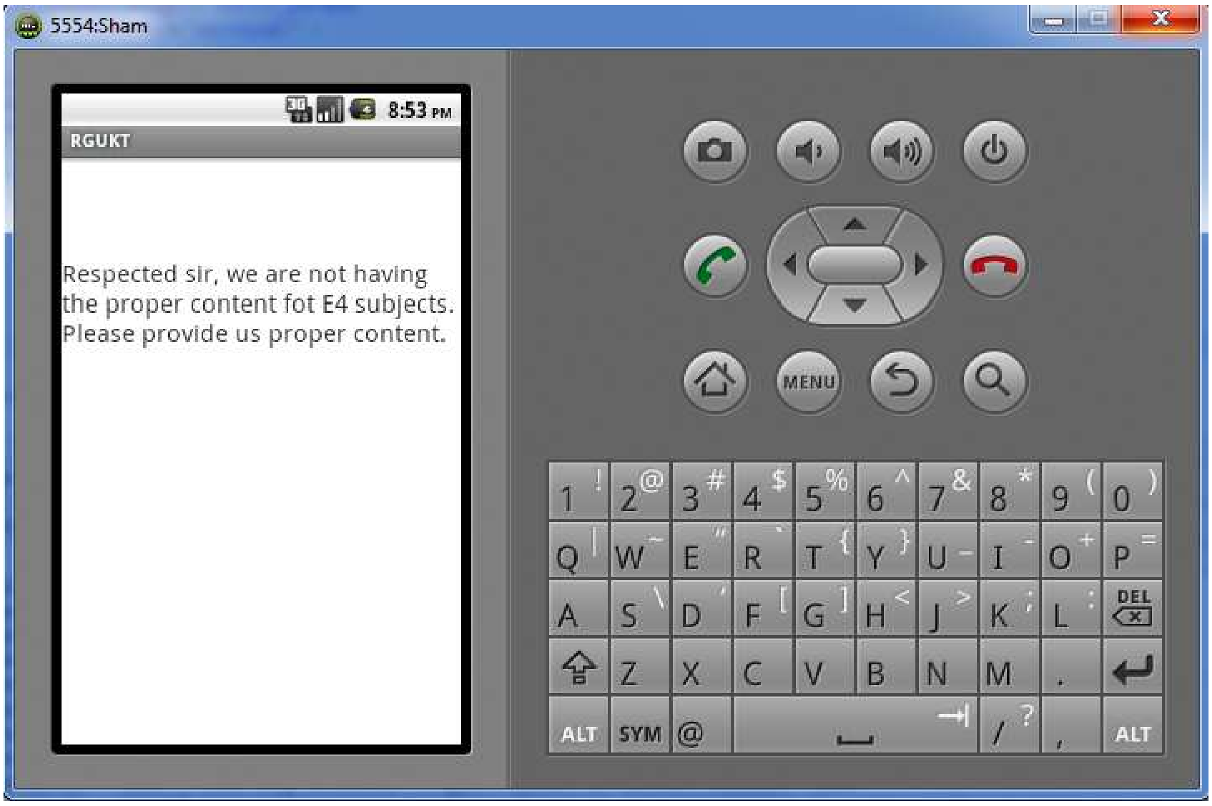


Figure 8:Menu



Respected sir, we are not having the proper content for E4 subjects. Please provide us proper contact.

Figure 9:Academicquerypost



Respected sir, we are not having the proper content fot E4 subjects. Please provide us proper content.

Figure 10:Displaying on Academic Coordinators profile

9 CONCLUSION

This is our first attempt in developing a mobile application which gave us a basic understanding of development and challenges of mobile application development. The main aim of this project is to implementing an app for android mobiles which will help for easy communication, understanding, between management and students

Document gives details about the building blocks of Android applications, Android application security and permissions. And also about Linux kernel which manages the memory, power and process management.

Android run time which includes the core libraries and DVM to run many applications at a time without affecting the other service function will be used.

10 REFERENCES

- http://developer.android.com/index.html
- http://www.codeproject.com/Articles/102065/Android-A-beginner-s-guide
- https://play.google.com/store/apps/details?id=com.threebanana.notes&hl=en
- http://www.cs.usfca.edu/~parrt/doc/java/JavaBasics-notes.pdf
- http://www.vogella.com/articles/Android/article.html
- http://www.androidhive.info/
- http://www.compiletimeerror.com/2013/07/android-tutorials.html
- http://www.javatechig.com/android-tutorials
- http://stackoverflow.com/