GetInfo using Android Application

A project Submitted to the Computer Science and Engineering Department, East West University In partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Engineering.

By

Md. Mehedi Hasan

2011-1-60-029

Mohammad imran

2011-1-60-032

Supervised By

Dr. Shaikh Muhammad Allayear

Assistant Professor

Department of Computer Science and Engineering, East West University

January 2016

ABSTRACT

GetInfo based on Android platform. It forethought and to make our life easy also saved time. By using this application anyone can easilly notify about his/her university activity like notice,result,calender etc. This report provides elaborate description on how we created the application and also how we eased the process of android development while creating the application.

Declaration

We hereby declare that this is submission is our own work and that to the best of our knowledge and belief it contains neither materials nor fact previously published or written by another person. Further, it does not contain material or fact which to a substantial extent has been accepted for the award of any degree of a university or any other institution of tertiary education except where an acknowledgment.

Signature of Candidate

………………………………

(Md. Mehedi Hasan)

………………………………

(Mohammad Imran)

Letter of Acceptance

The project entitled “GetInfo” submitted by Md.Mehedi Hasan, ID No. 2011-1-60-029 and Mohammad Imran, ID No. 2011-1-60-029, to the Department of Computer Science and Engineering, East West University, Dhaka-1212, Bangladesh is accepted by the Department for the partial fulfillment of requirements for the degree of Bachelor of Science in Computer Science and Engineering on January 14, 2015.

Dr. Shaikh Muhammad Allayear

Signature: ___________________________________

Date: _______________________________________

Assistant Professor, Department of Computer Science and Engineering,

East West University, Dhaka-1212, Bangladesh

Dr. Shamim Hasnat Ripon

Signature: ___________________________________

Date: _______________________________________

Chairperson and Associate Professor, Department of Computer Science and Engineering,

East West University Dhaka-1212, Bangladesh

Acknowledgements

The application "GetInfo" would never run successfully without the valuable encouragement and guidance from our supervisor Dr. Shaikh Muhammad Allayear, Assistant Professor, Department of Computer Science and Engineering, East West University. He enlightened, encouraged, and provided us with the ingenuity to transform our vision into reality. We particularly grateful to Dr. Shamim Hasnat Ripon, Chairperson and Associate Professor, Department of Computer Science and Engineering, East West University, for his encouragement. And also grateful to Siddhartha Sarkar for his guidance and counseling. We also grateful to all of our teachers and many of our friends who provided us with necessary software and suggestion to test the software as it developed. We would like to thank Google for creating android platform and keeping it free.

Table of Contents

Chapter 3: Proposed Models

Chapter 4: Implementation (Design)

Chapter 5: Conclusion and Future Work

Chapter 1: Introduction

1.1 Introduction

The world growing telecom network in the world with many users moving towards smart phones and majority by students. Study Life is one of many free options in the Android ecosystem and the app can integrate result, urgent notice, academic calendar, advising system and other events. These mobile applications provide a connection to the student with the university. It is very helpful to the student to connect with university about all activity. Android is open Google mobile platform which provide greater flexibility, Rapid Application Development Easy to Develop Interface with rich API collection. It mixture of Java, C++,PHP, HTML. Our creativity is to make an android based application that would be an efficient app for smart phone and also an entertaining app for user/students. So we started working to create an android application “Get Info” and that’s will make life became too easy.

1.2. Motivations

Usually in a university there is lot of activity as like as different kind of event, club etc. sometimes we cannot get any message or notification form university or from other sector. So it is difficult to get information from this source. We have "project name" apps to get the notification about urgent notice, semester result, academic calendar etc. Where if you have a smart phone and have an app which get notified you about the all of activity which are given up and it make the life easy and comfortable. With concerning all of those problems I have motivated to make this application.

1.3. Objectives

The main objectives of my application are:

- Registration and Login system
- Notification
- Updating and Show notification from database
- Advising System
- Evaluation
- Calendar
- showing all information in database

1.4. Contribution

In survey of existing mode we discuss the details about the android. We collect the all requirements to develop an android application. Software and hardware requirements are also discussed in the chapter to initialize an android development environment. In our project it discuss ion with details. We mention proposed model, database structure, class diagrams, class descriptions, data flow diagram, requirements, testing results, and requirements to run our application. At last we provide a user manual in Chapter User Manual that describes the proper way of using my application.

1.5 Organization of the Project Report

Following the step we reach the goal –

We collected the necessary information about Android.

We learned Android programming technique.

Collected requirements for our project.

We used SQLite database system for the application.

Design all necessary diagrams of our project Data flow diagram.

We tested our application and it passed in all the method We applied.

We created a manual for the general user.

Chapter 2

Literature Review and Survey of Existing Models

2.1. What is Android?

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. As of 2015, Android has the largest installed base of all operating systems.[1]

2.2 . History of Android

In this portion we will describe the history of android and I also show relation between my project and android.

2.2.1. Foundation

Android, Inc. was founded in Palo Alto, California, United States in October 2003 by Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc.), Nick Sears (once VP at T-Mobile), and Chris White (headed design and interface development at

WebTV) to develop, in Rubin's words "smarter mobile devices that are more aware of its owner's location and preferences". Despite the obvious past accomplishments of the founders and early employees, Android Inc. operated secretly, revealing only that it was working on software for mobile phones. That same year, Rubin ran out of money. Steve Perlman, a close friend of Rubin, brought him $10,000 in cash in an envelope and refused a stake in the company. [2]

2.2.2. Acquisition by Google

Google acquired Android Inc. on August 17, 2005, making Android Inc. a wholly owned subsidiary of Google. Key employees of Android Inc., including Andy Rubin, Rich Miner and Chris White, stayed at the company after the acquisition. Not much was known about Android Inc. at the time of the acquisition, but many assumed that Google was planning to enter the mobile phone market with this move.

2.2.3. Post-acquisition by Google

At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part. Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. Reports from the BBC and The Wall Street Journal noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google branded handset. Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

In September 2007, InformationWeek covered an Evacuee serve study reporting that Google had filed several patent applications in the area of mobile telephony.

2.2.4. Open Handset Alliance

On November 5, 2007, the Open Handset Alliance, a consortium of several companies which include Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, NVidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile and Texas Instruments unveiled itself. The goal of the Open Handset Alliance is to develop open standards for mobile devices. On the same day, the Open Handset Alliance also unveiled their first product, Android, a mobile device platform built on the Linux kernel version 2.6. On December 9, 2008, 14 new members joined, including ARM Holdings, Atheros Communications, Asustek Computer Inc., Garmin Ltd, Huawei Technologies, PacketVideo, Softbank, Sony Ericsson, Toshiba Corp, and Vodafone Group Plc. [10]

2.2.5. Android Open Source Project

The Android Open Source Project (AOSP) is led by Google, and is tasked with the maintenance and development of Android. According to the project "The goal of the Android Open Source Project is to create a successful real-world product that improves the mobile experience for end users." AOSP also maintains the Android Compatibility Program, defining an "Android compatible" device "as one that can run any application written by third-party developers using the Android SDK and NDK", to prevent incompatible Android implementations. The compatibility program is also optional and free of charge, with the Compatibility Test Suite also free and open-source.

2.2.6. Version history

Android has been updated frequently since the original release of "Astro", with each fixing bugs and adding new features. Each version is named in alphabetical order, with 1.5 "Cupcake" being the first named after a dessert and every update since following this naming convention. [10]

List of Android version names:

1. Cupcake
2. Donut
3. Eclair
4. Froyo
5. Gingerbread
6. Honeycomb
7. Ice Cream Sandwich
8. Android 4.2 Jelly Bean (API level 17)
9. Android 4.3 Jelly Bean (API level 18)
10. Android 4.4 Kit Kat (API level 19)
11. Android 5 Lollipop (API level 21)

2.3 Gingerbread refined the user interface, improved the soft keyboard and copy/paste features, improved gaming performance, SIP support (VoIP calls), and added support for Near Field Communication.

3.0 Honeycomb was a tablet-oriented release which supports larger screen devices and introduces many new user interface features, and supports multi core processors and hardware acceleration for graphics. The Honeycomb SDK has been released and the first device featuring this version, the Motorola Xoom tablet, went on sale in February 2011.

3.1 Honeycomb was announced at the 2011 Google I/O on 10 May 2011. One feature focuses on allowing Honeycomb devices to directly transfer content from USB devices.

3.2 Honeycomb released at July 15 2011, is "an incremental release that adds several new capabilities for users and developers". Highlights include optimization for a broader range of screen sizes; new "zoom-to-fill" screen compatibility mode; capability to load media files directly from the SD card; and an extended screen support API, providing developers with more precise control over the UI. Android 3.2 Honeycomb is the latest Android version that is available to tablets.

4.0.x Ice Cream Sandwich released at December 16, 2011, it's easy multitasking, rich notifications, customizable home screens, resizable widgets, and deep interactivity and adds powerful new ways of communicating and sharing.

4.1.x Jelly Bean released at July 9, 2012 Based on Linux kernel 3.0.31, Jelly Bean was an incremental update with the primary aim of improving the functionality and performance of the user interface. The performance improvement involved "Project Butter", which uses touch anticipation, triple buffering, and extended vsync timing and a fixed frame rate of 60 fps to create a fluid and "buttery-smooth" UI. Android 4.1 Jelly Bean was released to the Android Open Source Project on 9 July 2012, and the Nexus 7 tablet, the first device to run Jelly Bean.

4.2. x Jelly Bean released at November 13, 2012 its API level is 17.

4.3. x Jelly Bean released at July 24, 2013API level is 18.

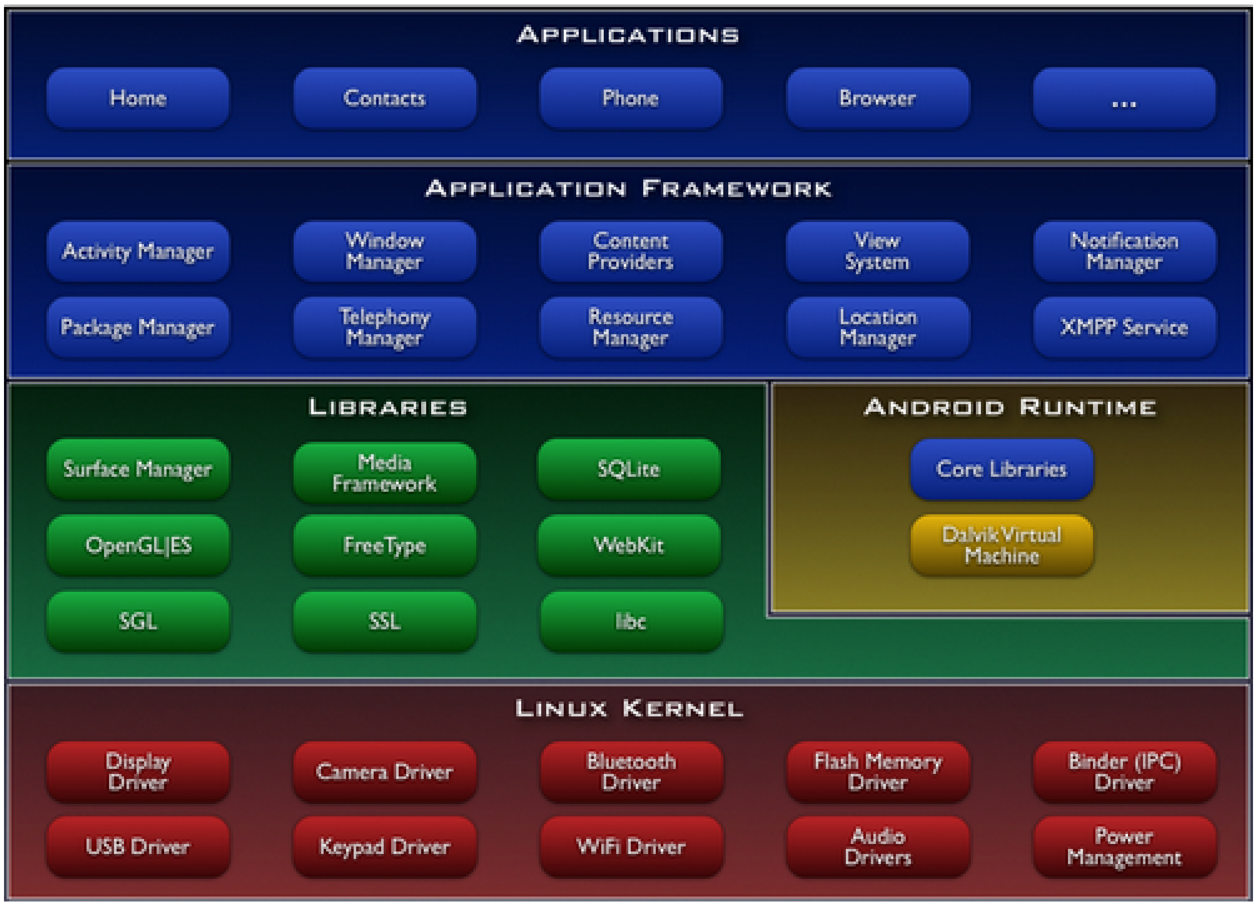
4.4 Kit Kat released at October 31, 2013 API level is 19. [4]

2.3 Design and Architecture of Android

Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvikdex-code (Dalvik Executable), which is usually translated from Java byte code.

The main hardware platform for Android is the ARM architecture. There is support for x86 from the Android x 86 projects and Google TV uses a special x86 version of Android.

2.4 Architecture of Android



2.5 Application

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.

Application Framework:

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

1. A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.

2. Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.

3. A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.

4. A Notification Manager that enables all applications to display custom alerts in the status bar.

5. An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack.

2.6 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.

Media Libraries - based on Packet Video’s Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.

Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.

Lib Web Core - a modern web browser engine which powers both the Android browser and an embeddable web view.

SGL - the underlying 2D graphics engine.

3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software pasteurizer.

Free Type - bitmap and vector font rendering.

SQLite- a powerful and lightweight relational database engine available to all applications.

2.7 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 Dalvik Executable (.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 the .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.

2.8 Applications

Applications are usually developed in the Java language using the Android Software Development Kit, but third party other development tools are available, including a Native Development Kit for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers and various cross platform mobile web applications frameworks. Applications can be acquired by end-users either through a store such as Google Play or the Amazon App store, or by downloading and installing the application's APK files from a site.

2.9 Google Play

Google Play is an online software store developed by Google for Android devices. An application program ("app") called "Play Store" is preinstalled on most Android devices and allows users to browse and download apps published by third-party developers, hosted on Google Play. As of February 2013, there were more than 800,000 apps available for Android, and the estimated number of applications downloaded from the Play Store exceeded 20 billion. The operating system itself is installed on 500 million total devices.

Only devices that comply with Google's compatibility requirements are allowed to preinstall and access the Play Store. The app filters the list of available applications to those that are compatible with the user's device, and developers may restrict their applications to particular carriers or countries for business reasons.

Google offers many free applications in the Play Store including Google Voice, Google Goggles, Gesture Search, Google Translate, Google Shopper, Listen and My Tracks. In August 2010, Google launched "Voice Actions for Android", which allows users to search, write messages, and initiate calls by voice

2.10 Security of Application

Android applications run in a sandbox, an isolated area of the operating system that does not have access to the rest of the system's resources, unless access permissions are granted by the user when the application is installed. Before installing an application, the Play Store displays all required permissions. A game may need to enable vibration, for example, but should not need to read messages or access the phonebook. After reviewing these permissions, the user can decide whether to install the application. The sandboxing and permissions system weakens the impact of vulnerabilities and bugs in applications, but developer confusion and limited documentation has resulted in applications routinely requesting unnecessary permissions, reducing its effectiveness.

The complexity of inter-application communication implies Android may have opportunities to run unauthorized code. Several security firms have released antivirus software for Android devices, in particular, Lookout Mobile Security, AVG Technologies, Avast!, F-Secure, Kaspersky, McAfee and Symantec. This software is ineffective as sandboxing also applies to such applications, limiting their ability to scan the deeper system for threats. A useful type of security applications program and service, often described as "Find My Phone", is available for Android, as well as for Microsoft Windows Phone and for Apple iPhone, whereby a registered user can find the approximate location of the phone, if switched on, over the Internet. This helps to locate lost or stolen phones. At least one of these can be installed on a phone after it has gone missing

2.11 Software Development Tools

In this portion we will describe about android app development tools and the way to development procedure.

2.11.1 Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse using the

Android Development Tools (ADT) Plug-in, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely). Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

2.11.2 Native Development Kit

Libraries written in C and other languages can be compiled to ARM or x86 native code and installed using the Android Native Development Kit. Native classes can be called from Java code running under the Dalvik VM using the System. Load Library call, which is part of the standard Android Java classes. Complete applications can be compiled and installed using traditional development tools. The ADB debugger gives a root shell under the Android Emulator which allows native ARM code or x 86 codes to be uploaded and executed. ARM or x 86 codes can be compiled using GCC on a standard PC. Running native code is complicated by the fact that Android uses a non-standard C library (libc, known as Bionic). The underlying graphics device is available as a frame buffer at /dev/graphics/fb0. The graphics library that Android uses to arbitrate and control access to this device is called the Skia Graphics Library (SGL), and it has been released under an open source license. Skia has backend for both win32 and UNIX, allowing the development of cross-platform applications, and it is the graphics engine underlying the Google Chrome web browser. Unlike Java App development based on the Eclipse IDE, the NDK is based on command-line tools and requires invoking them manually to build, deploy and debug the apps. Several third-party tools allow integrating the NDK into Eclipse and Visual Studio

2.11.3 Android Open Accessory Development Kit

The Android 3.1 platform (also back ported to Android 2.3.4) introduces Android Open Accessory support, which allows external USB hardware (an Android USB accessory) to interact with an Android-powered device in a special "accessory" mode. When an Android-powered device is in accessory mode, the connected accessory acts as the USB host (powers the bus and enumerates devices) and the Android-powered device acts as the USB device. Android USB accessories.

2.11.4 App Inventor for Android

On 12 July 2010, Google announced the availability of App Inventor for Android, a Web-based visual development environment for novice programmers, based on MIT's Open Blocks Java library and providing access to Android devices' GPS, accelerometer and orientation data, phone functions, text messaging, speech-to-text conversion, contact data, persistent storage, and Web services, initially including Amazon and Twitter. "We could only have done this because Android's architecture is so open," said the project director, MIT's Hal Abelson. Under development for over a year, the block-editing tool has been taught to non-majors in computer science at Harvard, MIT, Wellesley, Trinity College (Hartford,) and the University of San Francisco, where Professor David Wolber developed an introductory computer science course and tutorial book for non-computer science students based on App Inventor for Android

2.11.5 Hyper Next Android Creator

Hyper Next Android Creator (HAC) is a software development system aimed at beginner programmers that can help them create their own Android apps without knowing Java and the Android SDK. It is based on HyperCard that treated software as a stack of cards with only one card being visible at any one time and so is well suited to mobile phone applications that have only one window visible at a time. Hyper Next Android Creator's main programming language is simply called Hyper Next and is loosely based on HyperCard's Hyper Talk language. Hypertext is an interpreted English-like language and has many features that allow creation of Android applications. It supports a growing subset of the Android SDK including its own versions of the GUI control types and automatically runs its own.

2.12 The Simple Project

The goal of Simple is to bring an easy-to-learn-and-use language to the Android platform. Simple is a BASIC dialect for developing Android applications. It targets professional and nonprofessional programmers alike in that it allows programmers to quickly write Android applications that use the Android runtime components.

Similar to Microsoft Visual Basic 6, Simple programs are form definitions (which contain components) and code (which contains the program logic). The interaction between the components and the program logic happens through events triggered by the components. The program logic consists of event handlers which contain code reacting to the events. The Simple project is not very active, the last source code update being in August 2009.[7]

2.13. App Components

Like other application android application has its own components, below I will describe these components.

2.14. Application Fundamentals

Android applications are written in the Java programming language. The Android SDK tools compile the code-along with any data and resource files-into an Android package, an archive file with an .apk suffix.

All the code in a single .apk file is considered to be one application and is the file that Android powered devices use to install the application. Once installed on a device, each Android application lives in its own security sandbox:

The Android operating system is a multi-user Linux system in which each application is a different user.

By default, the system assigns each application a unique Linux user ID (the ID is used only by the system and is unknown to the application). The system sets permissions for all the files in an application so that only the user ID assigned to that application can access them.

Each process has its own virtual machine (VM), so an application's code runs in isolation from other applications.

By default, every application runs in its own Linux process. Android starts the process when any of the application's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other applications.

In this way, the Android system implements the principle of least privilege. That is, each application, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an application cannot access parts of the system for which it is not given permission. However, there are ways for an application to share data with other applications and for an application to access system services:

It's possible to arrange for two applications to share the same Linux user ID, in which case they are able to access each other's files. To conserve system resources, applications with the same user ID can also arrange to run in the same Linux process and share the same VM (the applications must also be signed with the same certificate). An application can request permission to access device data such as the user's contacts, SMS messages, the mountable

storage (SD card), camera, Bluetooth, and more. All application permissions must be granted by the user at install time.

That covers the basics regarding how an Android application exists within the system. The rest of this document introduces you to:

The core framework components that define your application.

The manifest file in which you declare components and required device features for your application.

Resources that are separate from the application code and allow your application to gracefully optimize its behavior for a variety of device configurations. [8]

2.15 Application Components

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

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

Components	**Description**
Activities	They dictate the UI and handle the user interaction to the smart phone screen

Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Activity:

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

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

```
public class MainActivity extends Activity {


}
```

Services

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

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

```
public class MyService extends Service {


}
```

Broadcast Receivers

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

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

```
public class MyReceiver  extends  BroadcastReceiver {


   public void onReceive(context,intent){}


}
```

Content Providers

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

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

```
public class MyContentProvider extends  ContentProvider {


   public void onCreate(){}


}
```

We will go through these tags in detail while covering application components in individual chapters.

Additional Components

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

Components	Description

Fragment	Represents a portion of user interface in an Activity.
Views	UI elements that are drawn on-screen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resource	External elements, such as strings, constants and drawable pictures.[9]

2.16 Activating Components

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an *intent*. Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your app or another.

An intent is created with an Intent object, which defines a message to activate either a specific component or a specific *type* of component—an intent can be either explicit or implicit, respectively.

For activities and services, an intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know). For example, an intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an Intent (for example, you can issue an intent to let the user pick a personal contact and have it returned to you—the return intent includes a URI pointing to the chosen contact).

For broadcast receivers, the intent simply defines the announcement being broadcast (for example, a broadcast to indicate the device battery is low includes only a known action string that indicates "battery is low").

The other component type, content provider, is not activated by intents. Rather, it is activated when targeted by a request from a ContentResolver. The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the ContentResolver object. This leaves a layer of abstraction between the content provider and the component requesting information (for security).

There are separate methods for activating each type of component:

- You can start an activity (or give it something new to do) by passing an Intent to startActivity() or startActivityForResult() (when you want the activity to return a result).
- You can start a service (or give new instructions to an ongoing service) by passing an Intent to startService(). Or you can bind to the service by passing an Intent to bindService().
- You can initiate a broadcast by passing an Intent to methods like sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().
- You can perform a query to a content provider by calling query() on a ContentResolver.

For more information about using intents, see the Intents and Intent Filters document. More information about activating specific components is also provided in the following documents: Activities, Services, BroadcastReceiver and Content Providers.[10]

2.17 The Manifest File

Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file (the "manifest" file). Your app must declare all its components in this file, which must be at the root of the app project directory.

The manifest does a number of things in addition to declaring the app's components, such as:

- Identify any user permissions the app requires, such as Internet access or read-access to the user's contacts.
- Declare the minimum API Level required by the app, based on which APIs the app uses.
- Declare hardware and software features used or required by the app, such as a camera, Bluetooth services, or a multitouch screen.

- API libraries the app needs to be linked against (other than the Android framework APIs), such as the Google Maps library.[10]

2.18 Declaring components

The primary task of the manifest is to inform the system about the application's components. For example, a manifest file can declare an activity as follows:

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

<manifest ... >

<applicationandroid:icon="@drawable/app_icon.png" ... >

<activityandroid:name="com.example.project.ExampleActivity"

android:label="@string/example_label" ... >

</activity>

...

</application>

</manifest>
```

In the <application> element, the android: icon attribute points to resources for an icon that identifies the application.

In the <activity> element, the android: name attribute specifies the fully qualified class name of the Activity subclass and the android: label attributes specifies a string to use as the user-visible label for the activity.

You must declare all application components this way:

<activity> elements for activities

<service> elements for services

<receiver> elements for broadcast receivers

<provider> elements for content providers

Activities, services, and content providers that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as Broadcast Receiver objects) and registered with the system by calling registerReceiver(). [10]

2.19 Declaring components capabilities

As discussed above, in Activating Components, you can use an Intent to start activities, services, and broadcast receivers. You can do so by explicitly naming the target component (using the component class name) in the intent. However, the real power of intents lies in the concept of intent actions. With intent actions, you simply describe the type of action you want to perform (and optionally, the data upon which you'd like to perform the action) and allow the system to find a component on the device that can perform the action and start it. If there are multiple components that can perform the action described by the intent, then the user selects which one to use. The way the system identifies the components that can respond to an intent is by comparing the intent received to the intent filters provided in the manifest file of other applications on the device.

When you declare a component in your application's manifest, you can optionally include intent filters that declare the capabilities of the component so it can respond to intents from other applications. You can declare an intent filter for your component by adding an <intent-filter> element as a child of the component's declaration element.

For example, an email application with an activity for composing a new email might declare an intent filter in its manifest entry to respond to "send" intents (in order to send email). An activity in your application can then create an intent with the ―send‖ action (ACTION_SEND), which the system matches to the email application's ―send‖ activity and launches it when you invoke the intent with startActivity(). [10]

2.20 Declaring application requirements

There are a variety of devices powered by Android and not all of them provide the same features and capabilities. In order to prevent your app from being installed on devices that lack features needed by your app, it's important that you clearly define a profile for the types of devices your app supports by declaring device and software requirements in your manifest file. Most of these declarations are informational only and the system does not read them, but external services such as Google Play do read them in order to provide filtering for users when they search for apps from their device.

For example, if your app requires a camera and uses APIs introduced in Android 2.1 (API Level 7), you should declare these as requirements in your manifest file like this:

```
<manifest ... >
   <uses-feature android:name="android.hardware.camera.any"
              android:required="true" />
   <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
   ...
</manifest>
```

Now, devices that do *not* have a camera and have an Android version *lower* than 2.1 cannot install your app from Google Play.

However, you can also declare that your app uses the camera, but does not *require* it. In that case, your app must set the required attribute to "false" and check at runtime whether the device has a camera and disable any camera features as appropriate.

More information about how you can manage your app's compatibility with different devices is provided in the Device Compatibility document.[10]

Device features

There are many hardware and software features that may or may not exist on a given Android powered device, such as a camera, a light sensor, Bluetooth, a certain version of OpenGL, or the fidelity of the touch screen. You should never assume that a certain feature is available on all Android-powered devices (other than the availability of the standard Android library), so you should declare any features used by your application with the <uses-feature> element.

Platform Version

Different Android-powered devices often run different versions of the Android platform, such as Android 1.6 or Android 2.3. Each successive version often includes additional APIs not available in the previous version. In order to indicate which set of APIs are available, each platform version specifies an API Level (for example, Android 1.0 is API Level 1 and Android 2.3 is API Level 9). If you use any APIs that were added to the platform after version 1.0, you should declare the minimum API Level in which those APIs were introduced using the <uses-sdk> element. It’s important that you declare all such requirements for your application, because, when you distribute your application on Google Play, the store uses these declarations to filter which applications are available on each device. As such, your application should be available only to devices that meet all your application requirements.

2.21 Application Resources

An Android application is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the application. For example, you should define animations, menus, styles, colors, and the layout of activity user interfaces with XML files. Using application resources makes it easy to update various characteristics of your application without modifying code and—by providing sets of alternative resources—enables you to optimize your application for a variety of device configurations (such as different languages and screen sizes). For every resource that you include in your Android project, the SDK build tools define a unique integer ID, which you can use to reference the resource from your application code or from other resources defined in XML. For example, if your application contains an image file named logo.png (saved in the res/drawable/ directory), the SDK tools generate a resource ID named R. drawable. logo, which you can use to reference the image and insert it in your user interface.[10]

One of the most important aspects of providing resources separate from your source code is the ability for you to provide alternative resources for different device configurations. For example, by defining UI strings in XML, you can translate the strings into other languages and save those

strings in separate files. Then, based on a language qualifier that you append to the resource directory's name (such as res/values-fr/ for French string values) and the user's language setting, the Android system applies the appropriate language strings to your UI.

Android supports many different qualifiers for your alternative resources. The qualifier is a short string that you include in the name of your resource directories in order to define the device configuration for which those resources should be used. As another example, you should often create different layouts for your activities, depending on the device's screen orientation and size. For example, when the device screen is in portrait orientation (tall), you might want a layout with buttons to be vertical, but when the screen is in landscape orientation (wide), the buttons should be aligned horizontally. To change the layout depending on the orientation, you can define two different layouts and apply the appropriate qualifier to each layout's directory name. Then, the system automatically applies the appropriate layout depending on the current device orientation.[11]

2.30 Database

A **database** is an organized collection of data.[1] It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A **database management system** (**DBMS**) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language.[*disputed – discuss*] Sometimes a DBMS is loosely referred to as a 'database'.

2.3.1 SQLite Database

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object. Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE
```

Database - Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username
VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");
```

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(1);
String password = resultSet.getString(2);   [14]
```

2.3.2 Website

A **website**, also written as **web site**,[1] or simply **site**,[2] is a set of related web pages typically served from a single web domain. A website is hosted on at least one web server, accessible via a network such as the Internet or a private local area network through an Internet address known as a uniform resource locator (URL). All publicly accessible websites collectively constitute the World Wide Web.

Web pages, which are the building blocks of websites, are documents, typically written in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML, XHTML). They may incorporate elements from other websites with suitable markup anchors. Webpages are accessed and transported with the Hypertext Transfer Protocol (HTTP), which may optionally employ encryption (HTTP Secure, HTTPS) to provide security and privacy for the user of the webpage content. The user's application, often a web browser, renders the page content according to its HTML markup instructions onto a display terminal.

The pages of a website can usually be accessed from a simple Uniform Resource Locator (URL) called the web address. The URLs of the pages organize them into a hierarchy, although hyperlinking between them conveys the reader's perceived site structure and guides the reader's navigation of the site which generally includes a home page with most of the links to the site's web content, and a supplementary about, contact and link page.

Some websites require a subscription to access some or all of their content. Examples of subscription websites include many business sites, parts of news websites, academic journal websites, gaming websites, file-sharing websites, message boards, web-based email, social networking websites, websites providing real-time stock market data, and websites providing various other services (e.g., websites offering storing and/or sharing of images, files and so forth).[15]

2.3.3 Website Design Feature

One of the elements of **good web design** is a lack of the elements that make bad web design. If you stay away from everything listed on the page about dorky web pages, you've probably got a pretty nice web site. In addition, keep these concepts in mind:

Text

Background does not interrupt the text

Text is big enough to read, but not too big

The hierarchy of information is perfectly clear

Columns of text are narrower than in a book to make reading easier on the screen

Navigation

Navigation buttons and bars are easy to understand and use

Navigation is consistent throughout web site

Navigation buttons and bars provide the visitor with a clue as to where they are, what page of the site they are currently on

Frames, if used, are not obtrusive

A large site has an index or site map

Links

Link colors coordinate with page colors

Links are underlined so they are instantly clear to the visitor

Graphics

Buttons are not big and dorky

Every graphic has an alt label

Every graphic link has a matching text link

Graphics and backgrounds use browser-safe colors

Animated graphics turn off by themselves

General Design

Pages download quickly

First page and home page fit into 800 x 600 pixel space

All of the other pages have the immediate visual impact within 800 x 600 pixels

Good use of graphic elements (photos, subheads, pull quotes) to break up large areas of text

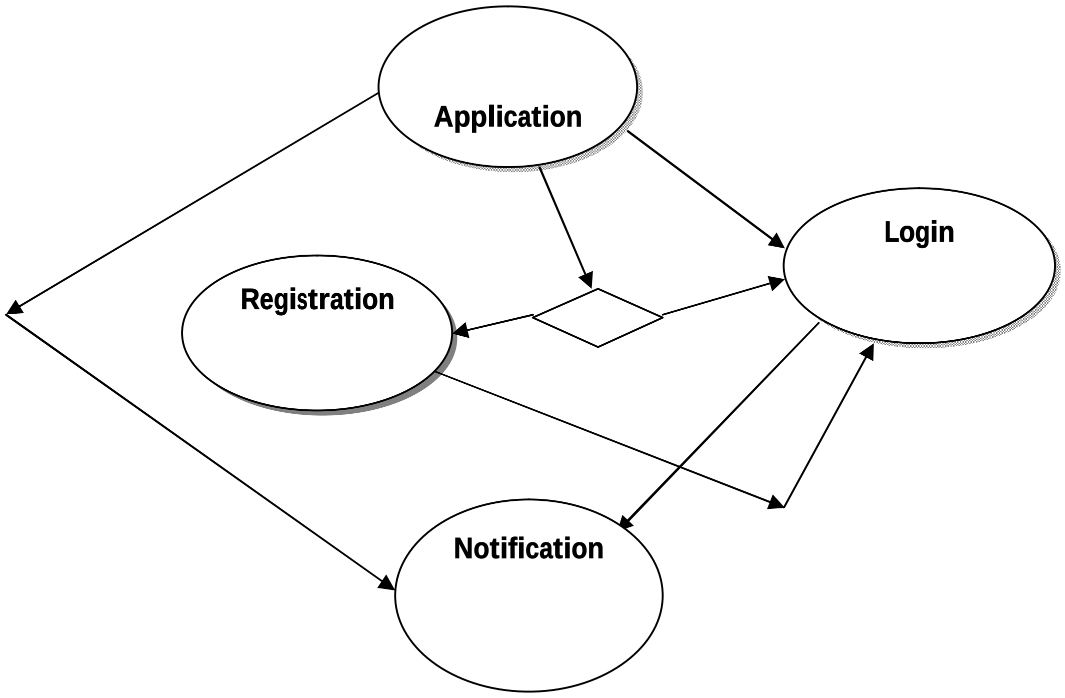
Every web page in the site looks like it belongs to the same site; there are repetitive elements that carry throughout the pages [16]

Chapter 3

Proposed Models

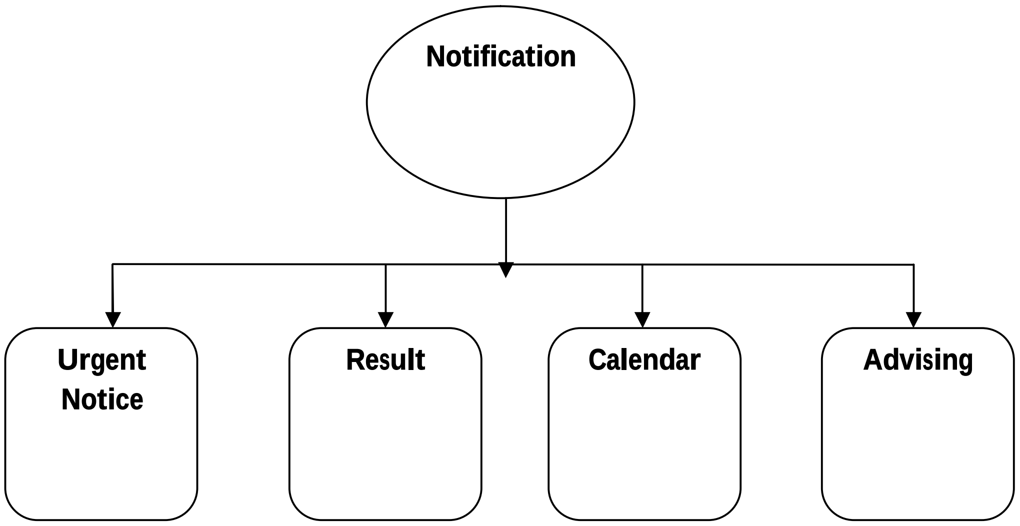
\

3.1 **Flow Chart for project**



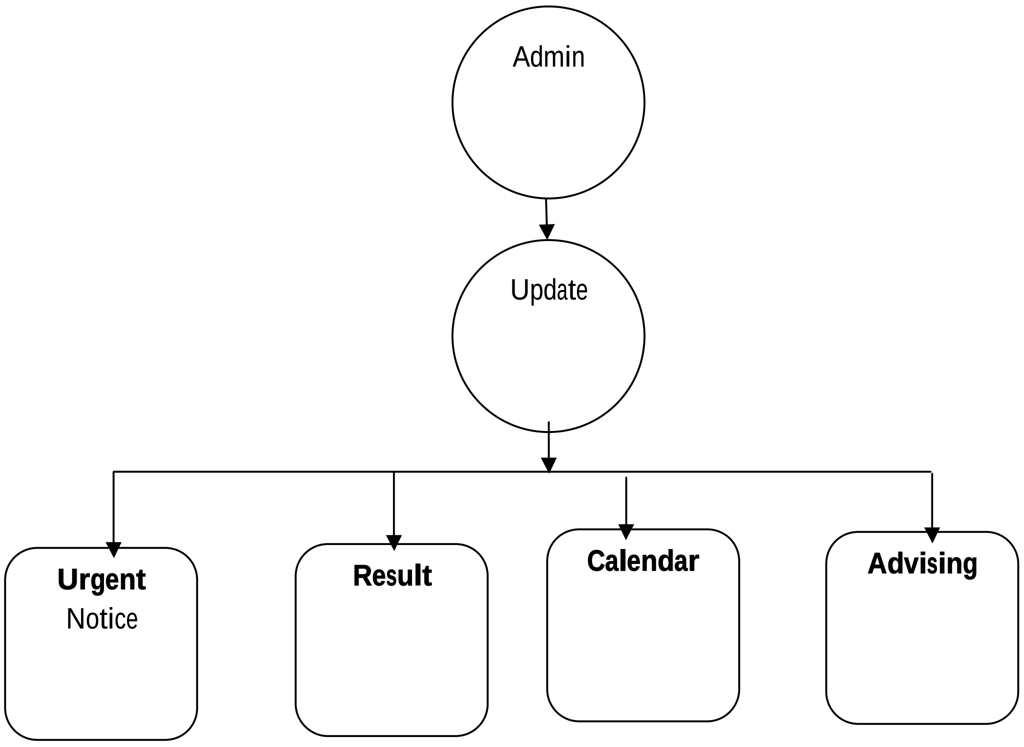
Flow chart for GetInfo android application

3.2 **Notification page**



Flow chart for notification page

3.3 **Admin Panel Page**



3.4 Implementation Procedure

Creating a fair environment for android development

Here are some simple prerequisite one must have to develop an android app.

Hardware Requirement Tools:

Development PC must be a fast one. we used a quad core machine clocked at @ 3.1 GHz with 6GB ram. A big monitor or two is also helpful. During debugging it really release the pain.

3.4.1 Android Development Environment

We are created android development environment for our project. Google basically supports the "Android Studio" version. But there are also other IDEs. We have used Android Studio for our development. There is also other software but we Android Studio because we wanted such IDE in which we could write java code and at the same time using the same IDE we could work on GUI for android. Android studio provides us lot of feature to build up our project. That’s why we use Android studio to develop our project

3.4.2 Project Setup:

For setup the android we go to "file'--->"new"---->"android project". After complete the direction you can get a simple project like 'hello'. Next procedure is "run" the project. Click "run"---->set window "emulator""------>"Target select". After complete the procedure you get a result.

3.4.3 Library Insertion:

For completing my project we need to add one library with our project. The library function is "useLibrary 'org.apache.http.legacy'[17]

To import this function we have to follow this procedure is File---> import--->Android---> apps-->build.gradle.

chapter 4

Implementation(Design)

4.1 Image View

4.1.1 Student Information

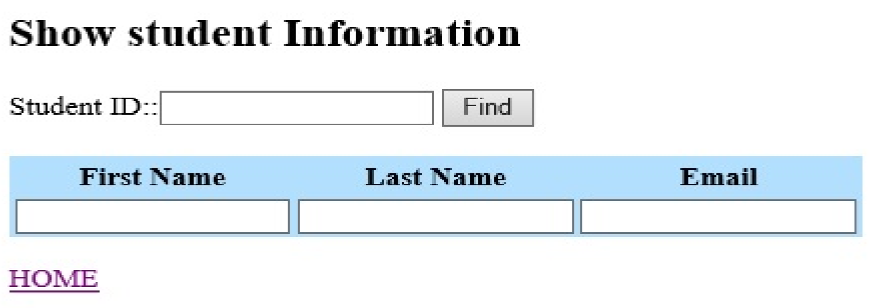


Fig: This web page will shows student Information. When students are registered.

4.1.2 Delete data

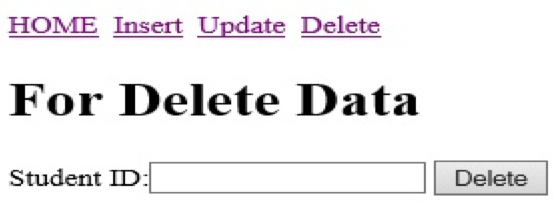


Fig: This web page will delete student result of the registered student.

4.1.3 Insert data

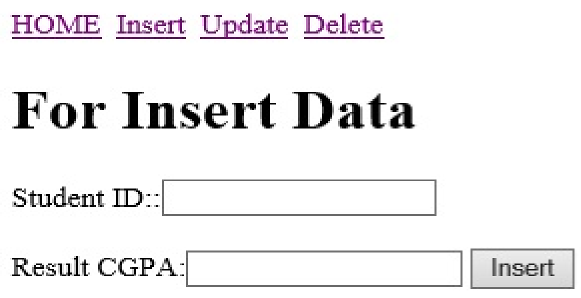


Fig: This web page will insert students result.

4.1.4 Update data

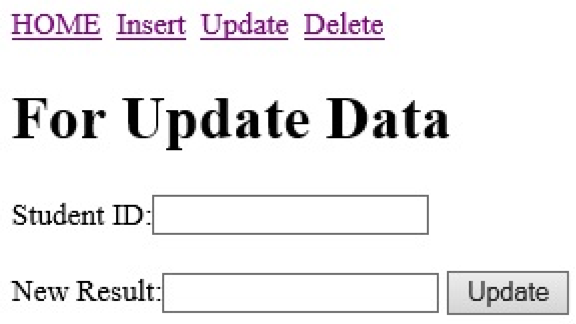


Fig: This web page will update students result.

4.1.5 Update Notice

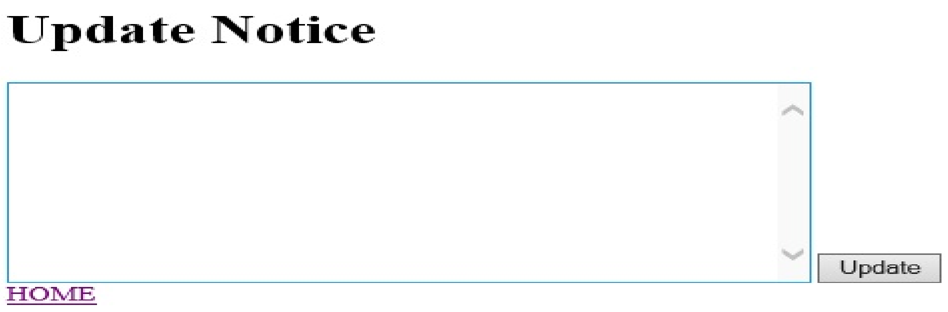
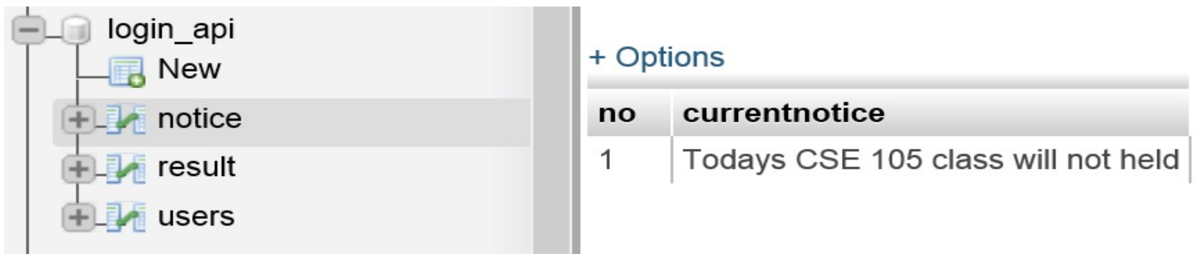


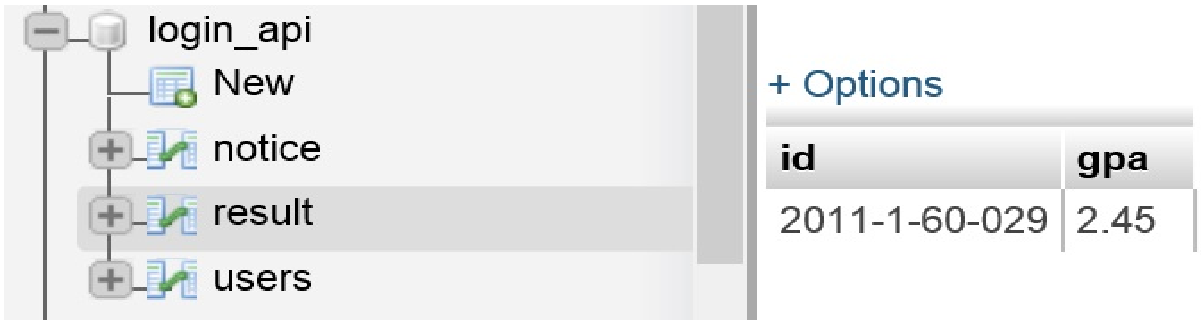
Fig: This web page will update Notice.

3.3.2 Database Image



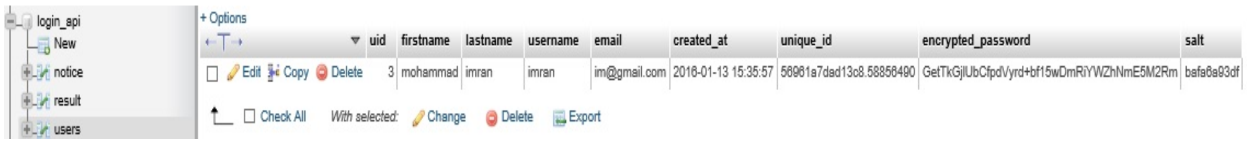
no	currentnotice
1	Todays CSE 105 class will not held

Fig: When admin insert Notice that time Database name “notice” will Update the notice.



id	gpa
2011-1-60-029	2.45

Fig: When admin insert or update or delete result that time this Database will work.



	uid	firstname	lastname	username	email	created_at	unique_id	encrypted_password	salt
Edit Copy Delete	3	mohammad	imran	imran	im@gmail.com	2016-01-13 15:35:57	56961a7dad13c8.58856490	GetTkGjlUbCfpdVyrd+bf15wDmRiYWZhNmE5M2Rm	bafa6a93df

Fig: When Users are registered that time this Database will work.

4.1.6 User Login

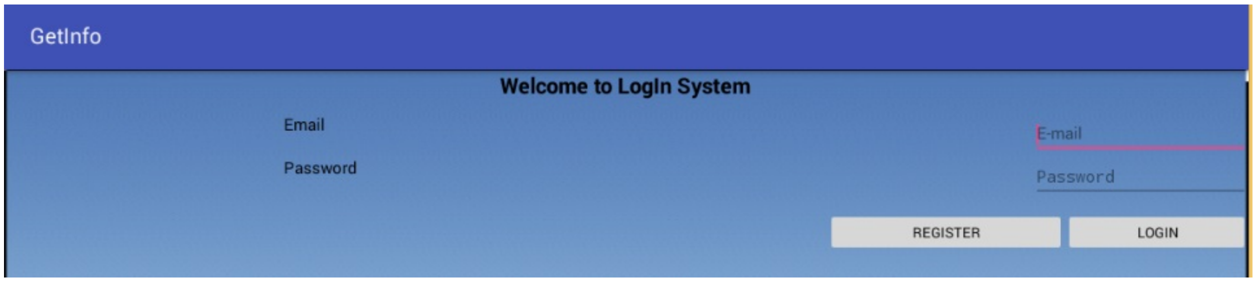
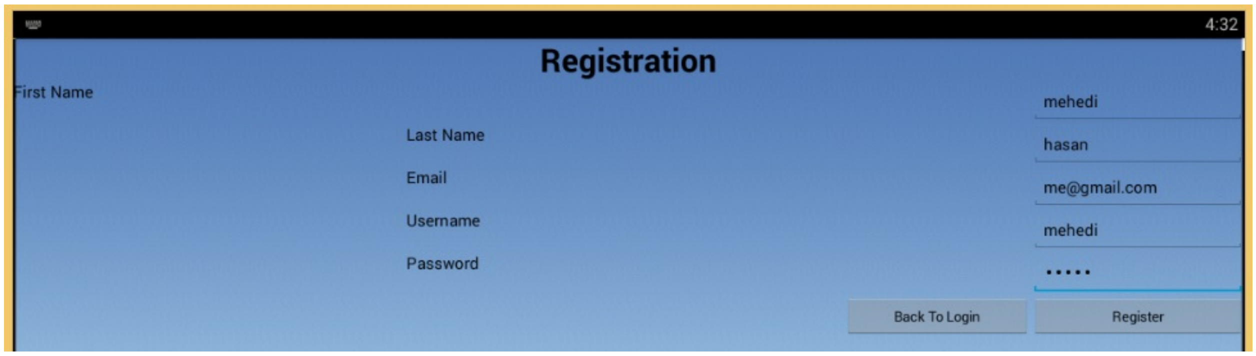


Fig: When user download this apps and run it, that time this layout will be show.

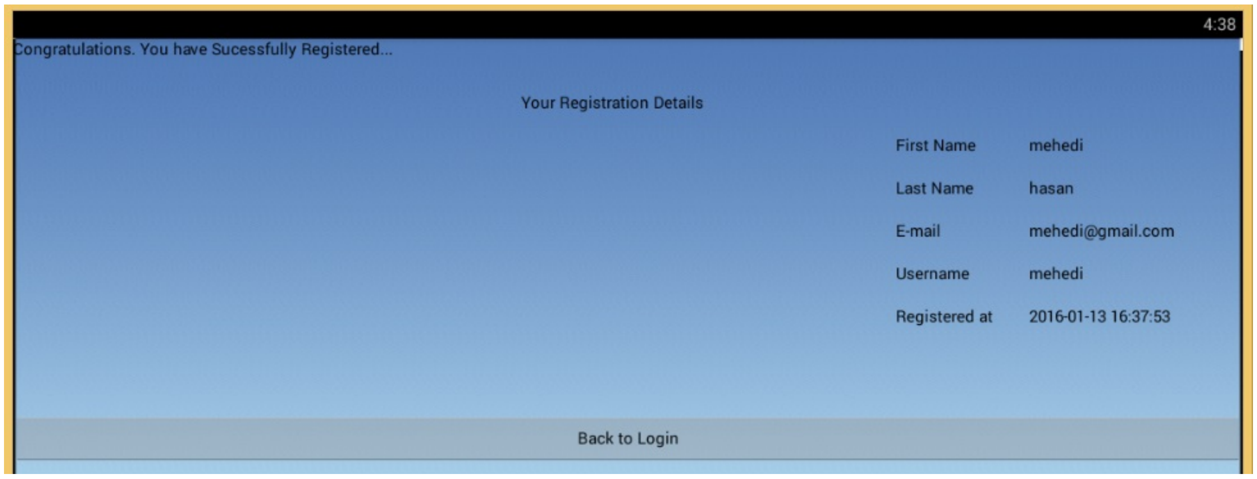
4.1.7 User Registration



First Name	mehedi
Last Name	hasan
Email	me@gmail.com
Username	mehedi
Password	••••••

Fig: When user presses the button register, that time this layout will be show.

4.1.8 User Registered



Congratulations. You have Sucessfully Registered...

Your Registration Details

First Name	mehedi
Last Name	hasan
E-mail	mehedi@gmail.com
Username	mehedi
Registered at	2016-01-13 16:37:53

Fig: When users are registered, that time this page will be show. And the registered user will get beck his information in this layout.

4.1.9 Registered user Login

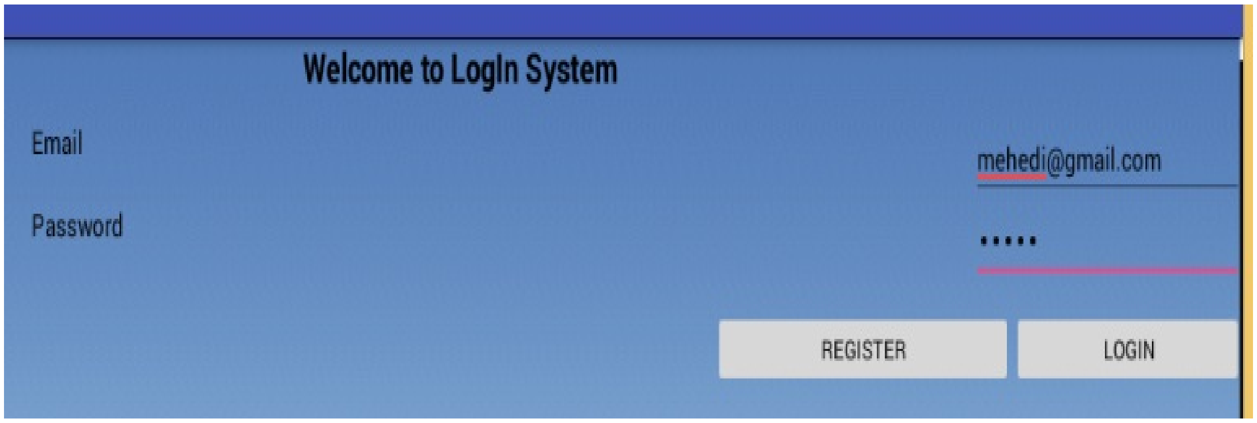


Fig: When users are registered and press the button LOGIN this layout will be show.

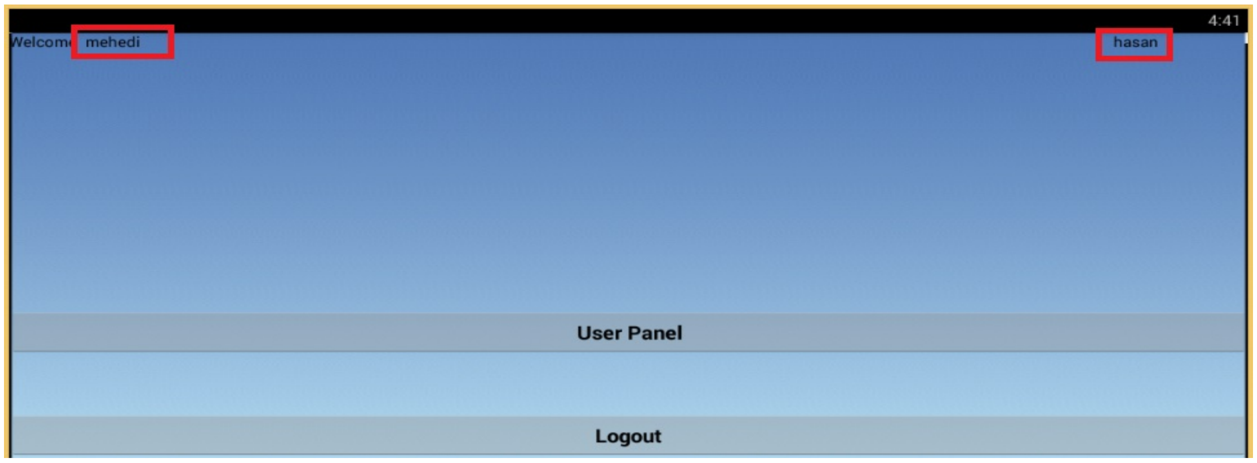
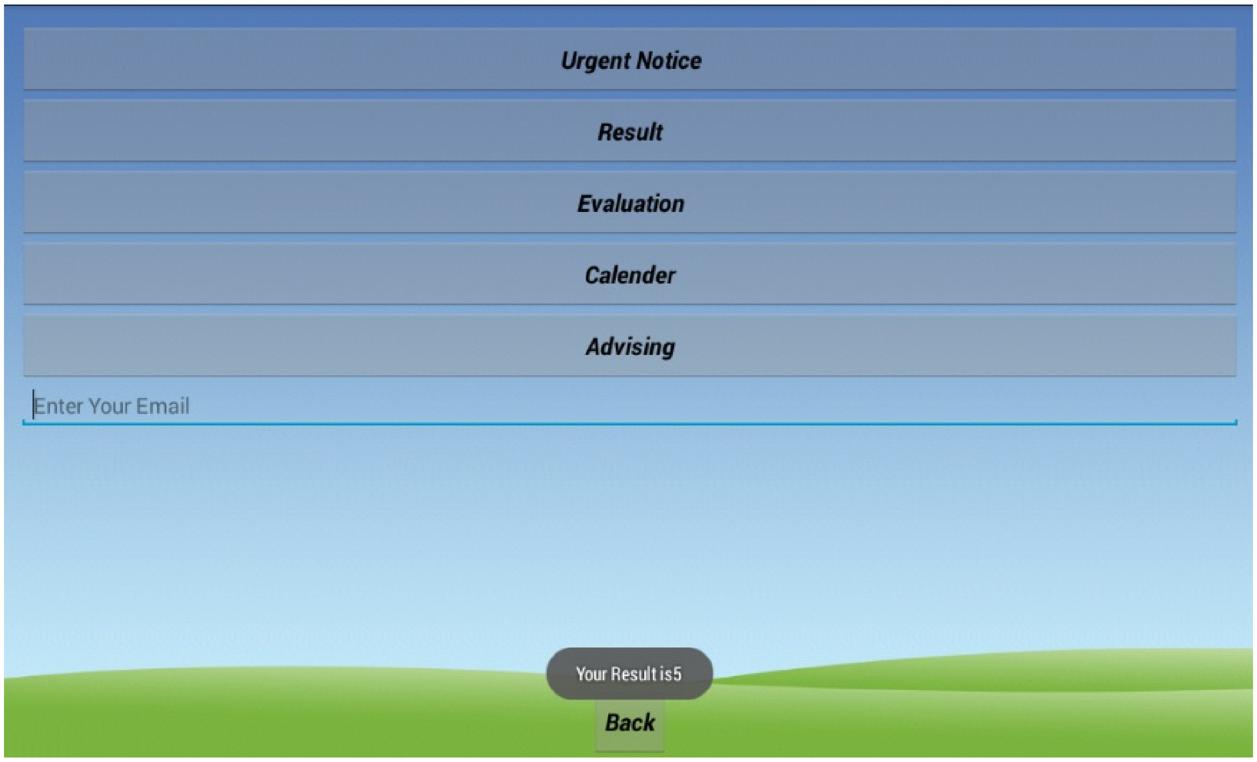


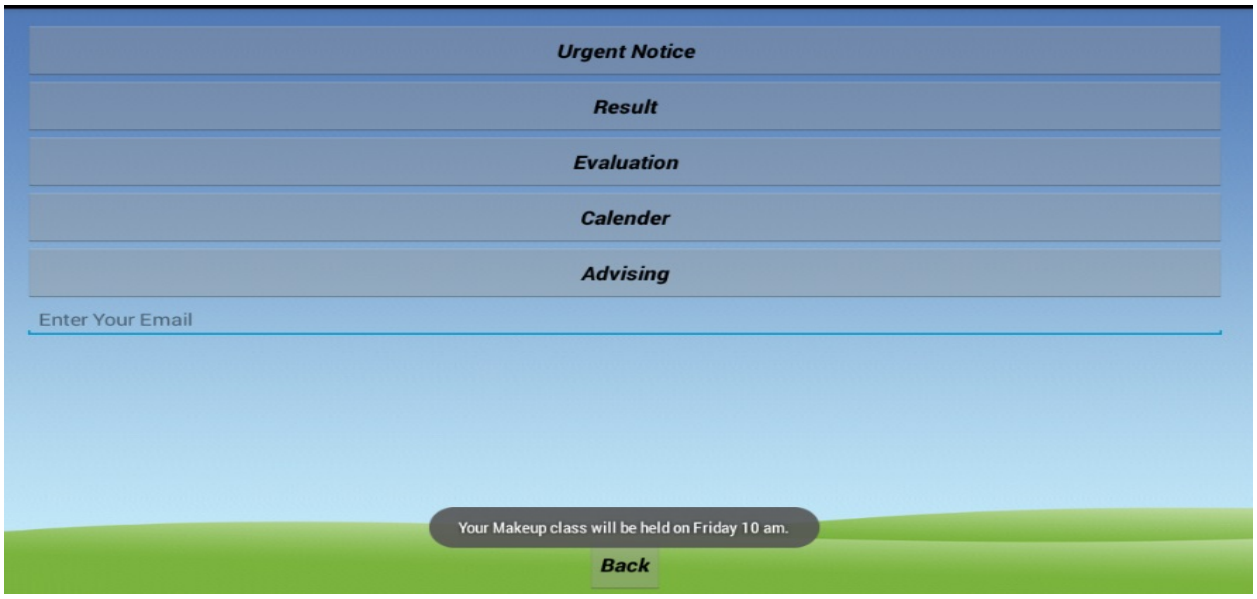
Fig: When users are registered give his correct EMAIL, PASSWORD, and button LOGIN this layout will be show. It is also a confirmation page.

4.1.10 User panel



Your Result is5

Fig: After successfully login and press the button USER_PANEL this user panel layout will be show. And press the button “result” the toast massage will display.



Your Makeup class will be held on Friday 10 am.

Fig: Press the button “urgent notice” this user panel layout will be show and the toast massage will display.

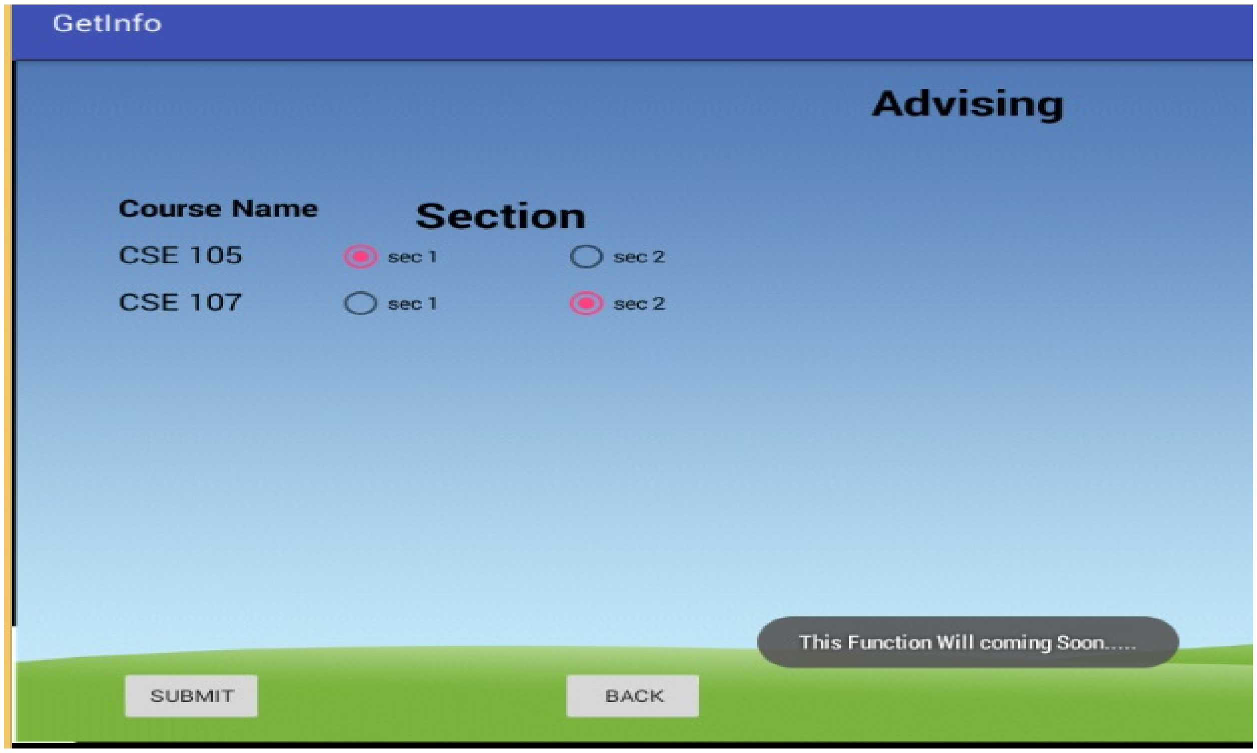
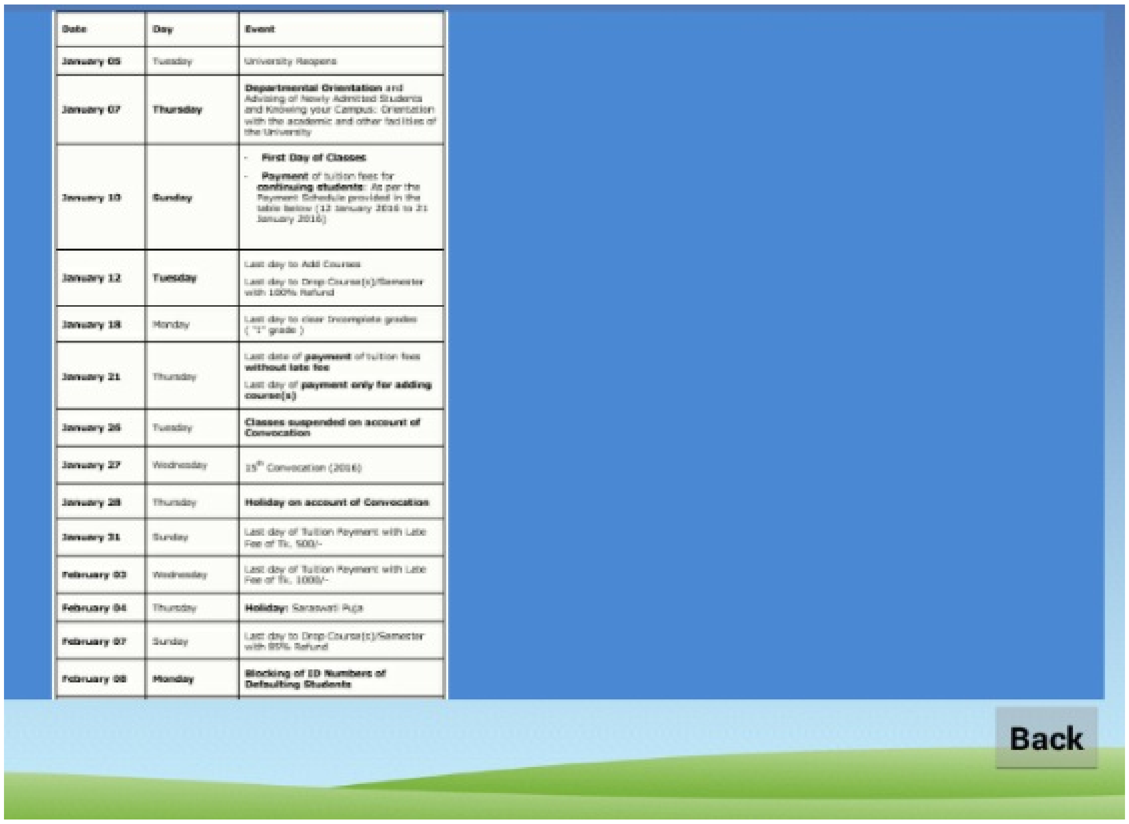


Fig: Press the button “advising” this user panel layout will be show and the toast massage will display.

Calendar



Date	Day	Event
January 05	Tuesday	University Reopens
January 07	Thursday	**Departmental Orientation** and Advising of Newly Admitted Students and Knowing your Campus: Orientation with the academic and other facilities of the University
January 10	Sunday	- **First Day of Classes** - **Payment** of tuition fees for **continuing students**: As per the Payment Schedule provided in the table below (12 January 2016 to 21 January 2016)
January 12	Tuesday	Last day to Add Courses Last day to Drop Course(s)/Semester with 100% Refund
January 18	Monday	Last day to clear Incomplete grades ("I" grade)
January 21	Thursday	Last date of **payment** of tuition fees **without late fee** Last day of **payment only for adding course(s)**
January 26	Tuesday	**Classes suspended on account of Convocation**
January 27	Wednesday	15th Convocation (2016)
January 28	Thursday	**Holiday on account of Convocation**
January 31	Sunday	Last day of Tuition Payment with Late Fee of Tk. 500/-
February 03	Wednesday	Last day of Tuition Payment with Late Fee of Tk. 1000/-
February 04	Thursday	**Holiday:** Saraswati Puja
February 07	Sunday	Last day to Drop Course(s)/Semester with 85% Refund
February 08	Monday	**Blocking of ID Numbers of Defaulting Students**

Fig: Press the button "calendar" this user panel layout will be show and this image will display.

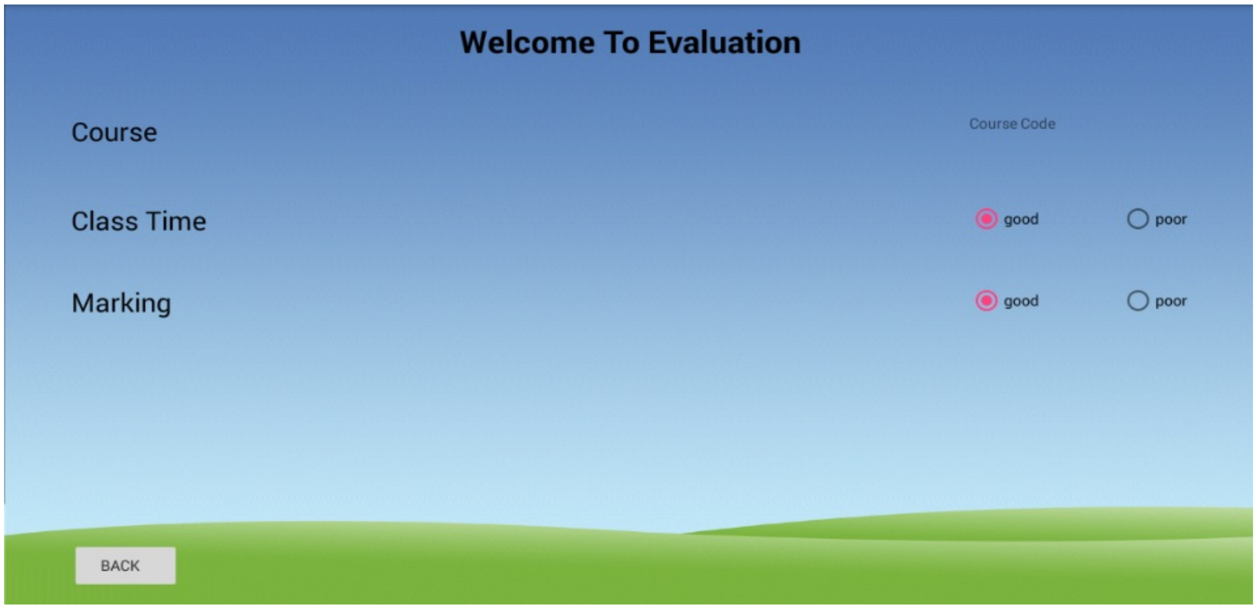


Fig: Press the button "evaluation" this user panel layout will display.

Chapter 5
Conclusion and Future Work

5.1 Conclusion:

Android was our choice because it is the most popular, user-friendly mobile operating system in the world. It is also most selling smart phone in the world .We think the application will be able to give the outcome that we wanted from the very beginning of our development process. In the process of developing the application we learned many things about the android operating system and the development related tricks. We also adopted easier and fresh ways, tricks, and techniques that would definitely help in future development. On the other hand it will help other android developers to develop it and modify it according to their way to make this application more fruitful and global. We will not say that we were perfect to make the application. We have also made a lot of mistakes. But we think this will help a lot to develop android application. "Get Info" using Android Application Very soon takes a place in the android market. And for sure some modification will come to make it useable globally. It is really very hard task to fulfill all the requirements with an application of smart phone. And we also try to contract with university for taking my apps. So there are a lot of chances of further development of the application in future. We have made my application keeping some space for further development.

5.2 Future Project Work:

We will improve our project work in future where you will able to check information from Notification. On the other hand by using the app we will try to make all administration system notification in every sector.

Reference

1. https://en.wikipedia.org/wiki/Android_(operating_system)

2. https://en.wikipedia.org/wiki/Android_(operating_system)#History

3 http://developer.android.com/guide/components/fundamentals.html

4 https://en.wikipedia.org/wiki/Android_version_history

5 http://forum.xda-developers.com/showthread.php?t=1595487

6 http://en.wikipedia.org/wiki/Android_(operating_system)

7http://developer.android.com/guide/components/fundamentals.html

8 http://en.wikipedia.org/wiki/Android_(operating_system)

9.http://www.tutorialspoint.com/android/android_application_component.htm
10..http://developer.android.com/guide/components/fundamentals.html

11. http://en.wikipedia.org/wiki/Google_Play

12. http://gaut.am/making-an-ocr-android-app-using-tesseract/

13. http://en.wikipedia.org/wiki/Optical_character_recognition

14. https://en.wikipedia.org/wiki/Database
15. https://en.wikipedia.org/wiki/Website
16. http://www.ratz.com/featuresgood.html
17. http://developer.android.com/about/versions/marshmallow/android-6.0-changes.html