

A Project Report  
on

<"QUIZO: An Android Quiz Game Application">

Submitted in partial fulfillment of the requirements for the Major Project II of

**Bachelor of Technology**  
in  
**Computer Science & Engineering**

Submitted by:

**Manilla Bhalla**  
500009172

**Neha Ghazi**  
500008959

Under the guidance  
**Dr.Ajay Shankar Singh**  
**Mr.Vishaal Kaushik**  
Designation  
**Asst.Prof.CIT,UPES**



**Department of Computer Science & Engineering**  
**COLLEGE OF ENGINEERING STUDIES**  
**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**  
**Dehradun- 248007**

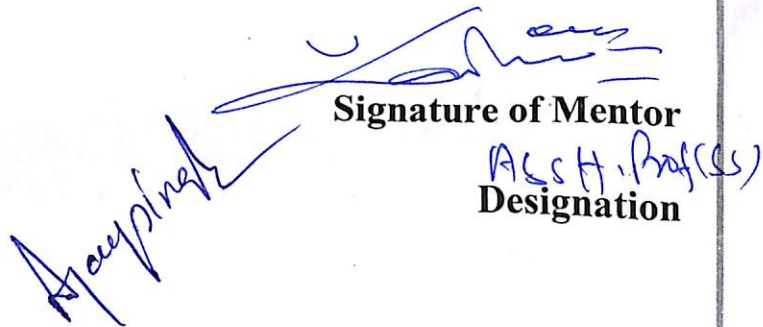
## CERTIFICATE

This is to certify that the Project entitled "**QUIZO: An Android Quiz Game Application**" submitted by

**Manilla Bhalla**  
**500009172**

**Neha Ghazi**  
**500008959**

for the partial fulfillment of the requirements of the course **Major Project II of Bachelor of Technology in Computer Science & Engineering** degree of **University of Petroleum & Energy Studies, Dehradun** embodies the confident work done by above students under my supervision.

  
**Signature of Mentor**  
**ALSH. Prof (CS)**  
**Designation**

## DECLARATION

We, Manilla Bhalla & Neha Ghazi bearing the Roll No: R780209017 and R780209043 respectively hereby declare that this Project work entitled "**QUIZO: An Android Quiz Game Application**" was carried out by us under the guidance and supervision of Mr. Ajay Shankar & Mr. Vishal Kaushik. This Project work is submitted to University of Petroleum & Energy Studies in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering during the Academic Semester Jan 2013 - Apr 2013. We also declare that, we have not submitted this dissertation work to any other university for the award of either degree or diploma.

Place: Dehradun

Manilla Bhalla  
Manilla Bhalla

Neha Ghazi

Neha Ghazi

Date: 15-04-2013

## ABSTRACT

### “ ABSTRACT OF YOUR PROJECT”

This android quiz game application will help the user to test their knowledge in technology.

Quiz game application contain multiple choice questions related to technology. If the user gives the correct answer then only the game will proceed then game will get over and the score will be displayed.

Android is a java based operating system that run on Linux 2.6 kernel and it is a software stack for mobile devices that include an operating system, middleware and key applications. Applications are usually developed in the Java language using the Android Software Development Kit. The Android SDK provides the tools necessary to begin developing applications on the Android platform using the Java programming language. Designed primarily for touch screen smart phones and tablet computers.

Features of an android include accelerated 3-D graphic engine (based on hardware support), database support powered by SQLite and an integrated web browser. Android also support XML-Based UI layout .

## **ACKNOWLEDGEMENT**

It is a pleasure to thank all those great many people who helped, supported and encouraged us during this project work.

Firstly we express our sincere gratitude to Mr Ajay Shankar Singh & Mr Vishal Kaushik, the guide of the project who carefully and patiently lent his valuable time and effort to give directions as well as to correct various documents with attention and care.

It is a great honour to do this project in this esteemed institution, and we would extend our thanks to the HOD, Prof Manish Prateek and other faculty members who have shared their vast knowledge and experience during our stay.

We do also like to appreciate the consideration of the Project Coordinator, our Faculties and colleagues, which enabled us to balance our work along with this project. It was their attitude that inspired us to do such an efficient and apposite work.

We are indebted to those people across the globe who have shared their knowledge and perspectives in the form of online tutorials, forums and other resources which helped us to a great extent whenever we met with technical obstacles during this endeavour.

We wish to avail this opportunity to express a sense of gratitude and love to all our friends and our family for their unwavering support, strength, help and in short for everything they have done during the crucial times of the progress of our project.

Last but not the least we thank GOD ALMIGHTY for HIS blessings and guidance without which this dream project wouldn't have been reality.

Manilla Bhalla

Neha Ghazi

# CONTENTS

---

Declaration	I
Abstract	ii
Acknowledgement	iii
Contents	iv

---

<b>1 Introduction</b>	<b>01 - 07</b>
1.1 What is Quizo?	01
1.2 Why Android?	02-09
	<b>10</b>
<b>2 Problem Definitions</b>	
<b>3 System Requirement Specification</b>	<b>11 - 18</b>
3.1 Overall Description	11
3.1.1 Product Perspective	11
3.1.2 Product Features	11-12
3.1.3 User Classes and Characteristics	12-15
3.1.4 Operating Environment	15
3.1.5 Design and Implementation Constraints	15-17
3.1.6 User Documentation	17-18

3.2	System Features	10-11
3.2.1	Functional Requirements	18
3.2.2	Non-Functional Requirements	19
<b>4</b>	<b>System Design</b>	<b>19-25</b>
	<b>What is UML Diagram</b>	
4.1	Use Case Diagram	19-22
4.2	Class Diagram	22-23
4.3	Sequence Diagram	23-24
4.4	Activity Diagram	24-25
<b>5.0</b>	<b>Code Implementation</b>	<b>26-44</b>
5.1	Android	26-30
5.2	Eclipse 3.4	30-
5.3	Overview of Java	30-34
5.4	Android SDK	34-35
<b>6.0</b>	<b>Testing</b>	<b>43-52</b>
6.1	Test Cases	
<b>7.0</b>	<b>Output Screens</b>	<b>52-67</b>
<b>8</b>	<b>Conclusion</b>	<b>68</b>
<b>9</b>	<b>Bibliography</b>	<b>68-</b>
<b>10</b>	<b>Appendix</b>	<b>69-73</b>

## 1.1 What is QUIZO?

---

This android quiz game application will help the user to test their knowledge in computers.

Quiz game application contain multiple choice questions related to computers. If the user gives the correct answer then only the game will proceed then game will get over and the score will be displayed.

Android is a java based operating system that run on Linux 2.6 kernel and it is a software stack for mobile devices that include an operating system, middleware and key applications.

Applications are usually developed in the Java language using the android kit ie

Android Software Development Kit. The Android SDK provides the tools necessary to begin

developing applications on the Android platform using the Java programming language.

Designed primarily for touch screen smart phones and tablet computers.

Features of an android include accelerated 3-D graphic engine (based on hardware support),

database support powered by SQLite and an integrated web browser. Android also support

XML-Based UI layout .



## 1.2 Why Android?

---

Android is a freely downloadable open source software stack for mobile devices that includes an operating system, middleware and key applications based on Linux and Java.

Android is a Java based operating system that runs on Linux 2.6 kernel and it is a software stack for mobile devices that includes an operating system, middleware and key applications.

Applications are usually developed in the Java language using the Android Software Development Kit.

The Android SDK provides the tools necessary to begin developing applications on the Android platform using the Java programming language. Designed primarily for touch screen smart phones and tablet computers.

### 1.2.1 Features of Android

- **Handset layouts** Android can adapt to traditional smart phone layouts, as well as other VGA, 2D, and 3D graphics libraries.
- **Storage** Android uses SQLite to store all its junk-- I mean, information.
- **Connectivity** Android supports a wide variety of technologies, including Bluetooth, WiFi, GSM/EDGE, and EV-DO.
- **Messaging** MMS and SMS are available for Android, as well as threaded text messaging. So you can send as many text messages as you like.
- **Web Browser** Android comes pre-loaded with the Web Kit application. Remember if

you don't like it, you can always switch it out for something else later on thanks to the open source nature of the Google Android backend.

- **Java Virtual Machine** Software you write in Java can be compiled in Dalvik Byte codes (say that five times fast. I keep ending up with "Danish light bulb".) These can then be put into a Dalvik Virtual Machine. Basically more robust applications are supported than on some other Mobile Operating Systems.
- **Media Support** Android supports a wide range of audio, video, media, and still formats. MPEG-4, OGG, and AAC are just a few of these. Unfortunately the Media Player as its known right now is pretty basic, although more robust offerings on are the horizon from 3<sup>rd</sup> Party developers.
- **Additional Hardware Support** Got a touch screen you want to put to its full use? No problem. Android is capable of utilizing outside hardware like GPS, accelerometers, and all that other fun stuff.

## 1.2.2 Building blocks to an Android application

There are four building blocks to an Android application:

- **Activity**
- **Broadcast Intent Receiver**
- **Service**
- **Content Provider**

### 1. Activity

Activities are the most common of the four Android building blocks. An activity is usually a single screen in your application. Each activity is implemented as a single class that extends the **Activity**

**base class.** Your class will display a user interface composed of Views and respond to events. Most applications consist of multiple screens. For example, a text messaging application might have one screen that shows a list of contacts to send messages to, a second screen to write the message to the chosen contact, and other screens to review old messages or change settings. Each of these screens would be implemented as an activity. Moving to another screen is accomplished by starting a new activity. In some cases an activity may return a value to the previous activity -- for example an activity that lets the user pick a photo would return the chosen photo to the caller.

## **2.Intent and Intent Filters**

Android uses a special class called Intent to move from screen to screen. Intent describes what an application wants done. The two most important parts of the intent data structure are the action and the data to act upon. Typical values for action are MAIN (the front door of the application), VIEW, PICK, EDIT, etc. The data is expressed as a URI. For example, to view contact information for a person, you would create intent with the VIEW action and the data set to a URI representing that person.

There is a related class called an Intent Filter. While an intent is effectively a request to do something, an intent filter is a description of what intents an activity (or Broadcast Receiver, see below) is capable of handling. An activity that is able to display contact information for a person would publish an Intent Filter that said that it knows how to handle the action VIEW when applied to data representing a person.

Activities publish their Intent Filters in the **AndroidManifest.xml** file. The new activity is informed of the intent, which causes it to be launched. The process of resolving intents happens at run time when start

Activity is called, which offers two key benefits:

- Activities can reuse functionality from other components simply by making a request in the form
- of an Intent
- Activities can be replaced at any time by a new Activity with an equivalent Intent Filter

### 3. BROADCAST INET RECEIVER

You can use a Broadcast Receiver when you want code in your application to execute in reaction to an external event, for example, when the phone rings, or when the data network is available, or when it's midnight. Broadcast Receivers do not display a UI, although they may use the Notification Manager to alert the user if something interesting has happened. Broadcast Receivers are registered in `AndroidManifest.xml`, but you can also register them from code using **`Context.registerReceiver ()`**.

Your application does not have to be running for its BroadcastReceivers to be called; the system will start your application, if necessary, when a BroadcastReceiver is triggered. Applications can also send their own intent broadcasts to others with **`Context.sendBroadcast ()`**.

### 4. SERVICE

A **Service** is code that is long-lived and runs without a UI. A good example of this is a media player playing songs from a play list. In a media player application, there would probably be one or more activities that allow the user to choose songs and start playing them. However, the music playback itself should not be handled by an activity because the user will expect the music to keep playing even after navigating to a new screen. In this case, the media player activity could start a service using **`Context.startService ()`** to run in the background to keep the music going. The system will then keep the music playback service running until it has finished. Note that you can connect to a service (and start it if it's not already running) with the **`Context.bindService ()`** method. When connected to a service, you can communicate with it through an interface exposed by the service. For the music service, this might allow you to pause, rewind, etc.

### 5. CONTENT PROVIDER

Applications can store their data in files, an SQLite database, or any other mechanism that makes sense. A content provider, however, is useful if you want your application's data to be shared with other applications. A content provider is a class that implements a standard set of methods to let other applications store and retrieve the type of data that is handled by that content provider.

Not every application needs to have all four, but your application will be written with some combination of these.

All the components needed for android application should listed in an xml file called

AndroidManifest.xml. This is an XML file where you declare the components of your application and what their capabilities and requirements are.

### **1.2.3 Storing, Retrieving and Exposing Data**

A typical desktop operating system provides a common file system that any application can use to store and read files that can be read by other applications. Android uses a different system on Android, all application data are private to that application. However, Android also provides a standard way for an application to expose its private data to other applications. This section describes the many ways that an application can store and retrieve data, expose its data to other applications, and also how you can request data from other applications that expose their data.

Android provides the following mechanisms for storing and retrieving data:

#### **1. Preferences**

A lightweight mechanism to store and retrieve key/value pairs of primitive data types. This is typically used to store application preferences.

#### **2. Files**

You can store your files on the device or on a removable storage medium. By default, other applications cannot access these files.

#### **3. Databases**

The Android APIs contain support for SQLite. Your application can create and use a private SQLite database. Each database is private to the package that creates it.

#### **4. Content Providers**

A content provider is a optional component of an application that exposes read/write access

to an application's private data, subject to whatever restrictions it wants to impose. Content providers implement a standard request syntax for data, and a standard access mechanism for the returned data. Android supplies a number of content providers for standard data types, such as personal contacts.

## **5.Network**

Don't forget that you can also use the network to store and retrieve data.

## **1.2.4 Security and Permissions in Android**

Android is a multi-process system, where each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data.

### **1.System Architecture**

A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user. This includes reading or writing the user's private data such as contacts or e-mails, reading or writing another application's files, performing network access, keeping the device awake, etc.

An application's process is a secure sandbox. It can't disrupt other applications, except by explicitly declaring the permissions it needs for additional capabilities not provided by the basic sandbox. These permissions it requests can be handled by the operating in various ways, typically by automatically allowing or disallowing based on certificates or by prompting the user. The permissions required by an application are declared statically in that application, so they can be known up-front at install time and will not change after that.

## 2.Application Signing

All Android applications (.apk files) must be signed with a certificate whose private key is held by their developer. This certificate identifies the author of the application. The certificate does *not* need to be signed by a certificate authority: it is perfectly allowable, and typical, for Android applications to use self-signed certificates. The certificate is used only to establish trust relationships between applications, not for wholesale control over whether an application can be installed. The most significant ways that signatures impact security is by determining who can access signature-based permissions and who can share user IDs.

## 3.User IDs and File Access

Each Android package (.apk) file installed on the device is given its own unique Linux user ID, creating a sandbox for it and preventing it from touching other applications (or other applications from touching it). This user ID is assigned to it when the application is installed on the device, and remains constant for the duration of its life on that device.

## 4.Using Permissions

A basic Android application has no permissions associated with it, meaning it can not do anything that would adversely impact the user experience or any data on the device. To make use of protected features of the device, you must include in your AndroidManifest.xml one or more **<uses-permission>** tags declaring the permissions that your application needs.

The permissions provided by the Android system can be found at **Manifest.permission**. Any application may also define and enforce its own permissions, so this is not a comprehensive list of all possible permissions.

A particular permission may be enforced at a number of places during your program's operation:

- At the time of a call into the system, to prevent an application from executing certain functions.
- When starting an activity, to prevent applications from launching activities of other applications. Both sending and receiving broadcasts, to control who can receive your broadcast or who can send a broadcast to you.
- When accessing and operating on a content provider.
- Binding or starting a service

## 5.Declaring and Enforcing Permissions

To enforce your own permissions, you must first declare them in your AndroidManifest.xml using one or more **<permission>** tags.

The **<protection Level>** attribute is required, telling the system how the user is to be informed of applications requiring the permission, or who is allowed to hold that permission, as described in the linked documentation.

The **<permission Group>** attribute is optional, and only used to help the system display permissions to the user. You will usually want to set this to either a standard system group (listed in `android.Manifest.permission_group`) or in more rare cases to one defined by yourself. It is preferred to use an existing group, as this simplifies the permission UI shown to the user.

Note that both a label and description should be supplied for the permission. These are string resources that can be displayed to the user when they are viewing a list of permissions (`android:label`) or details on a single permission (`android:description`). The label should be short, a few words describing the key piece of functionality the permission is protecting. The description should be a couple sentences describing what the permission allows a holder to do. Our convention for the description is two sentences, the first describing the permission, the second warning the user of what bad things can happen if an application is granted the permission.



## 1.2.5 Applications Developed on Android Platforms

- In September 2008, Motorola confirmed that it was working on hardware products that would run Android.
- Huawei Technologies is planning to launch smart phones that would run Android in Q1 2009.
- Lenovo is working on an Android-based mobile phone that supports the Chinese 3G TD-SCDMA standard.
- HTC is planning a "portfolio" of Android based phones to be released summer of 2009. 8
- Sony Ericsson is planning to release an Android based handset in the summer of 2009.
- Samsung plans to offer a phone based on Google's Android operating system in the second quarter of 2009.
- GiiNii Movit Mini is a Internet device based on Google's Android operating system

## CHAPTER 2 PROBLEM DEFINITION

This android quiz game application will help the user to test their knowledge in technology.

Quiz game application contain multiple choice questions related to technology. If the user gives the correct answer then only the game will proceed then game will get over and the score will be displayed.

## CHAPTER 3 SYSTEM REQUIREMENT SPECIFICATION

### 3.1 Overall Description

---

### **3.1.1 Product Perspective**

Quizo is an android application that can be used on a client system, so that it can be accessed in a mobile and can be used to check the technical knowledge.

### **3.1.2 Product Features**

The major functionalities of the proposed system are:

- User-Friendly environment which can enhance interest of the user.
- The application can be adjusted on any mobile which have android operating system .
- Sound-effects in the application gives it professional touch
- No Internet access is required to play the application.
- Android is "openness." The promise is that developers can produce applications without interference.
- "The fact that (Android) is an advanced, open operating system is important to the development community, but customers don't buy operating systems.
- The interface is flexible.

- The Android Platform provides a rich security model that allows developers to request the capabilities, or access, needed by their application and to define new capabilities that other applications can request.
- Developers have full access to the same framework APIs used by the core applications

### **3.1.2.1 ASSUMPTIONS AND DEPENDENCIES**

- Having only hardware is not sufficient, to access an application Software is must.
- Assumption is made in such a way that the mobile is charged with enough battery.
- The battery should be in a working mode.
- The one who using the mobile must have a minimum knowledge of how to play the game.

### **3.1.3 User Classes and Characteristics**

The class of users that we aim to serve with this product is diverse and diverged on how they are going to use it. The users must have minimal knowledge of the android operating system.

We also aim this product to be useful to the students of college or professional.

### **3.1.4 Operating Environment**

The product is built for Android environment and the internet is not required to play the application. The framework required are-

- 1) **Java Run Time Environment(JRE)**
- 2) **eclipse ( eclipse-SDK-3.6.2-win32**
- 3) **JDK (cnet2\_jdk-6-windows-i586)**

### ***JDK contents***

The JDK has as its primary components a collection of programming tools, including:

- **appletviewer** – this tool can be used to run and debug Java applets without a web browser
- **apt** – the annotation-processing tool<sup>[4]</sup>
- **extcheck** – a utility which can detect JAR-file conflicts
- **idlj** – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- **java** – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.

- **javac** – the Java compiler, which converts source code into Java bytecode
- **javadoc** – the documentation generator, which automatically generates documentation from source code comments
- **jar** – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- **javah** – the C header and stub generator, used to write native methods
- **javap** – the class file disassembler
- **javaws** – the Java Web Start launcher for JNLP applications
- **JConsole** – Java Monitoring and Management Console
- **jdb** – the debugger
- **jhat** – Java Heap Analysis Tool (experimental)
- **jinfo** – This utility gets configuration information from a running Java process or crash dump. (experimental)
- **jmap** – This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)

- **jps** – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
- **jrunscript** – Java command-line script shell.
- **jstack** – utility which prints Java stack traces of Java threads (experimental)
- **jstat** – Java Virtual Machine statistics monitoring tool (experimental)
- **jstatd** – jstat daemon (experimental)
- **keytool** – tool for manipulating the keystore
- **pack200** – JAR compression tool
- **policytool** – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
- **VisualVM** – visual tool integrating several command-line JDK tools and lightweight<sup>*clarification needed*</sup> performance and memory profiling capabilities
- **wsimport** – generates portable JAX-WS artifacts for invoking a web service.
- **xjc** – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

The JDK also comes with a complete Java Runtime Environment, usually called a *private* runtime, due to the fact that it is separated from the "regular" JRE and has extra contents. It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

## **Ambiguity between a JDK and an SDK**

The JDK forms an extended subset of a software development kit (SDK). In the descriptions that accompany its recent releases, which implement Java SE, EE and ME, Sun acknowledges that under its terminology, the JDK forms the subset of the SDK which has the responsibility for the writing and running of Java programs.† The remainder of the SDK comprises extra software, such as application servers, debuggers, and documentation

### **4) Android Development Tools Plug-in(SDK manager and AVD manager) in ECLIPSE.**

#### **3.1.5 Design and Implementation Constraints**

(a-) Design for first page include background image ,name

Quiz game ie Quizo and three Buttons-

**Start, Scores, Instructions**

1-)Start –by clicking on the start button the quiz game will start.

2-)Scores-include the score

3-)Instructions-include the instructions (how to play the game).

(b-) Design for second page include multiple choice question

It include four options (means four options) the user has to click on the right answer.

## **3.2 System Features**

---

### **3.2.1 Functional Requirements**

The system is required to perform the following functions.

1. Display all the information about the application that is being developed and some set of instructions the user might want to remember before he sets up the system for configuring global time.
2. Sign in your application with jar signer before running your application
3. Install your apk file with android bridge (i.e adb) .
4. After executing your application
5. Ability to rotate the shapes in clockwise and anti clockwise direction.
6. Use up and down arrow keys for rotating.
7. Able to display score



8. Ability to pause and resume and stop the application

### **3.2.2 Non-Functional Requirements**

- 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).
- SQLite for structured data storage.
- GPS
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE.
- The system is expected to run on low memory devices also.
- The system should not consume lot of bandwidth so that the other applications will block for the internet.
- The system should provide secured access to the web server.

### **Requirements Specifications**

#### **3.2.3 Software Requirements:**

J2SE, Android, Linux, Windows XP, Eclipse 3.4, Mobile IDE Plug-in.

### **3.2.4 Hardware Requirements:**

Pentium IV with 2GHZ, 1 GB RAM, 40 GB Hard Drive, Android Phone (optional).

## **CHAPTER 4**

## **SYSTEM DESIGN**

### **4.1 What is UML?**

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non- software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

### **4.2 Goals of UML:**

**The primary goals in the design of the UML were:**

- Provide users with a ready-to-use, expressive visual modeling language so they can develop

and exchange meaningful models.

- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components. Integrate best practices.

### **4.3 Why Use UML?**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these

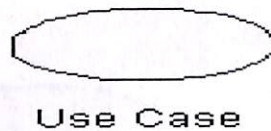
architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

#### 4.4 UML Diagrams:

UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include.

#### 4.5 Use case Diagram:

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

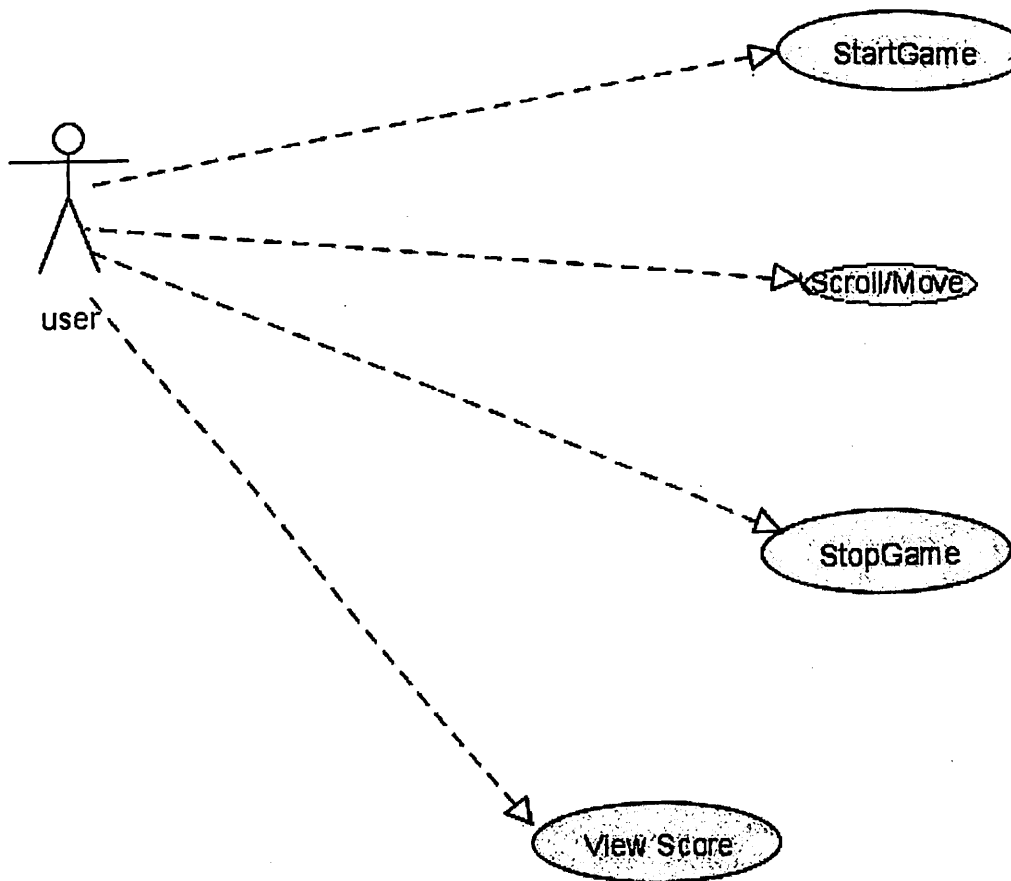


An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

#### When to Use: Use Cases Diagrams:

Use cases are used in almost every project. They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project

continues more might become visible.



#### 4.6 Class Diagram:

Class diagrams are widely used to describe the types of objects in a system and their relationships.

Class diagrams model class structure and contents using design elements such as classes, packages

and objects. Class diagrams describe three different perspectives when designing a system,

conceptual, specification, and implementation. These perspectives become evident as the diagram is

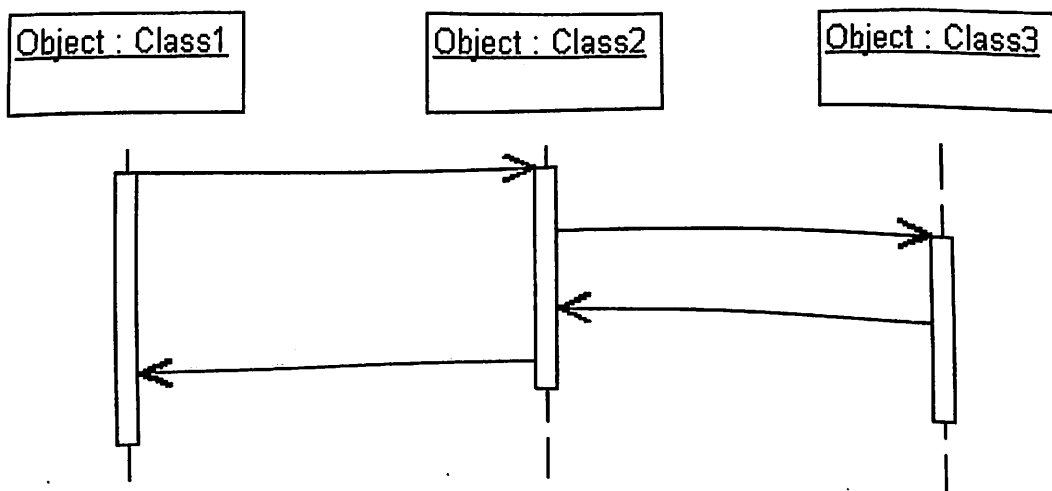
created and help solidify the design. This example is only meant as an introduction to the UML and class diagrams.

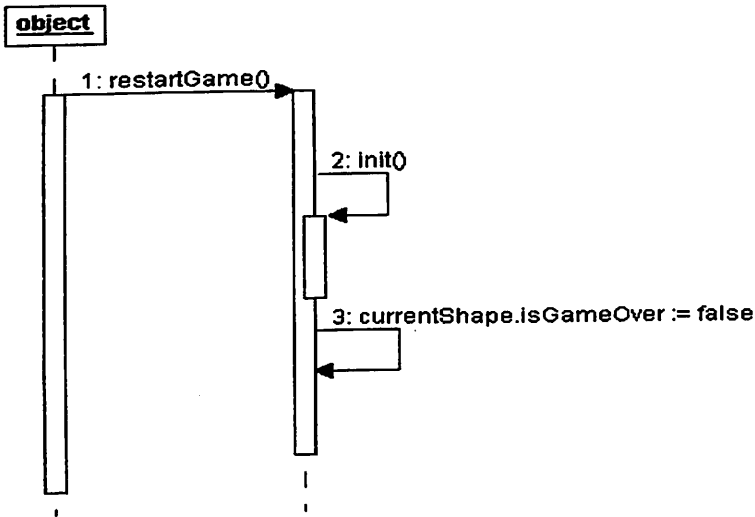
### When to Use: Class Diagrams:

Class diagrams are used in nearly all Object Oriented software designs. Use them to describe the Classes of the system and their relationships to each other.

### 4.7 Sequence Diagram:

Sequence diagrams demonstrate the behavior of objects in a use case by describing the objects and the messages they pass. The diagrams are read left to right and descending. The example below shows an object of class 1 start the behavior by sending a message to an object of class 2. Messages pass between the different objects until the object of class 1 receives the final message.





## 4.8 Activity Diagram:

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

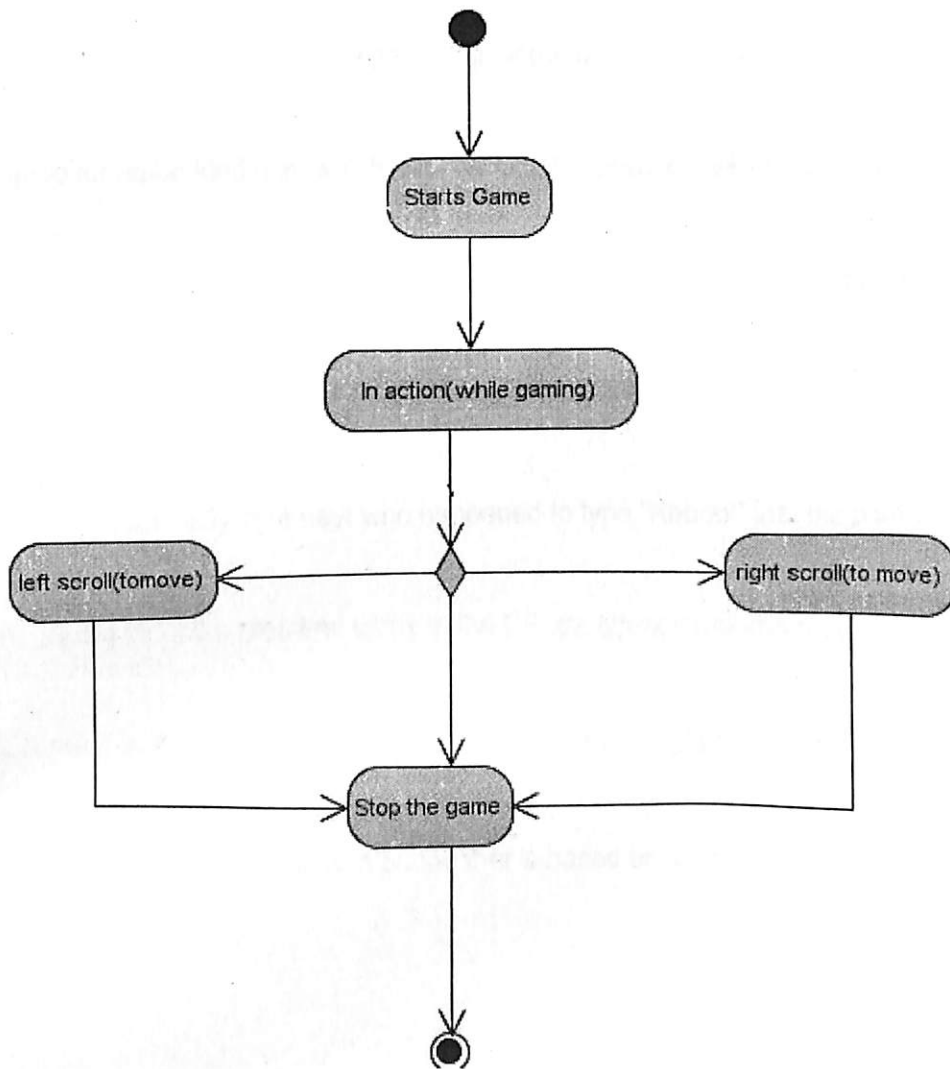
### When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analyzing a use

case by describing what actions needs to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.

However, activity diagrams should not take the place of interaction diagrams and state diagrams.

Activity diagrams do not give detail about how objects behave or how objects collaborate.





## 5.1 Android:

The Android platform is a software stack for mobile devices including an operating system, middleware and key applications. Developers can create applications for the platform using the Android SDK.

Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use, which runs on top of a Linux kernel. An embarrassing bug found on the G1 Phone has been fixed by Google. After starting up the phone if a user then typed "reboot", the phone would reboot.

The bug was found accidentally by a user who happened to type "Reboot" into his phone.

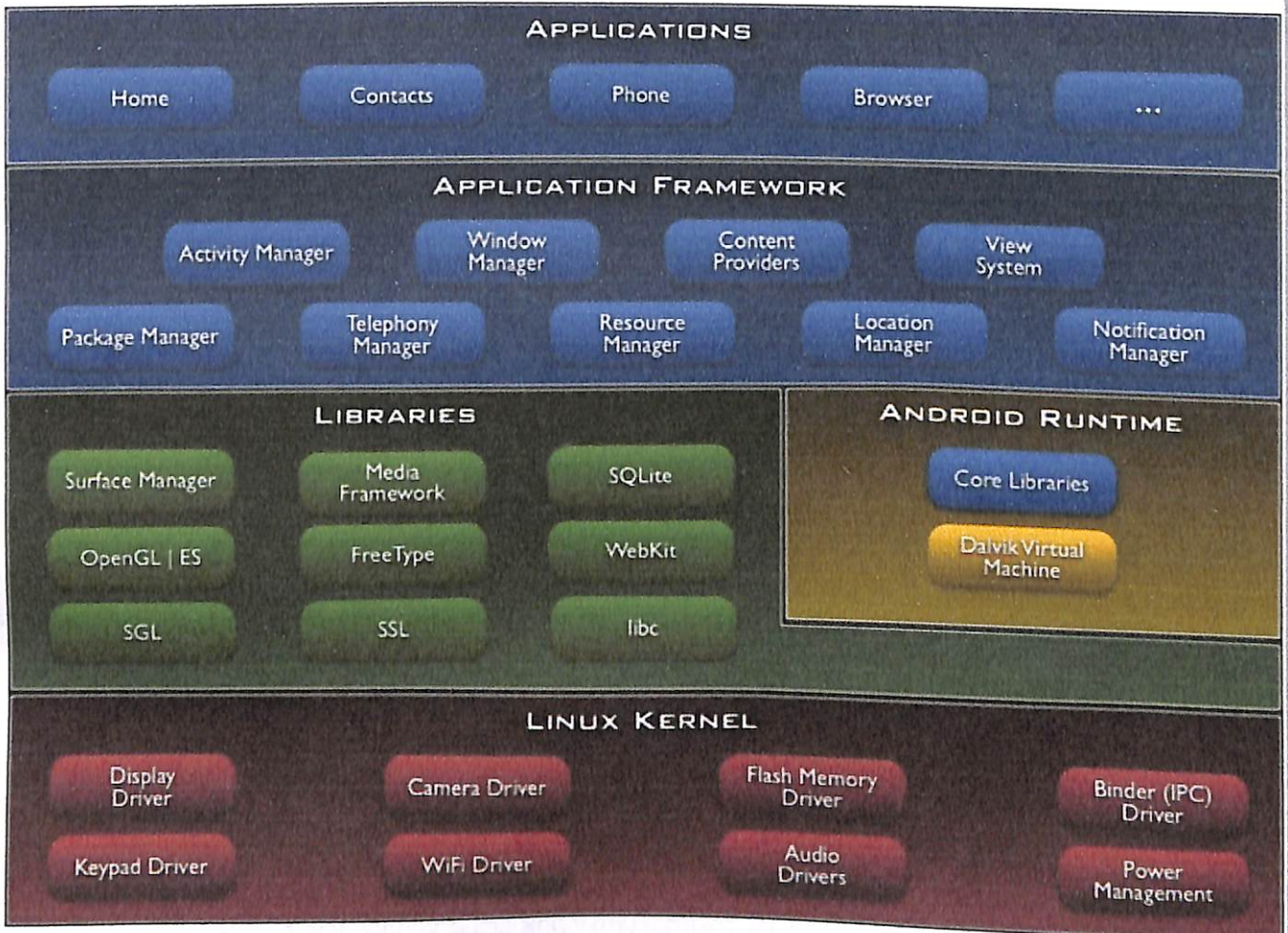
Google moved quickly to fix the problem, users in the US are already reporting having received the update and according to the BBC, users in the UK will have the update by 12th November.

The G1 has had surprisingly few bugs for a phone that is based on a completely new platform and is the first phone to use it.

### 5.1.1 Android Architecture:

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

## 5.1.1.1 Applications:



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

- 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

- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all applications to display custom alerts in the status bar
- An Activity Manager that manages the life cycle of applications and provides a common navigation back stack

### **5.1.1.3 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

- LibWebCore - 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 rasterizer
- FreeType - bitmap and vector font rendering
- SQLite - a powerful and lightweight relational database engine available to all applications

#### **5.1.1.4 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.

#### **5.1.1.5 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 the software stack.

#### **5.2 Eclipse 3.4:**

An open-source Java IDE and platform for rich client applications Eclipse is an open source platform-independent software framework for delivering what the project calls or is known as "rich-client applications" (as opposed to "thin clients", this means the clients perform heavy-duty work on the host.

So far this framework has typically been used to develop Ides (Integrated Development Environments), such as the highly-regarded Java IDE called Java Development Toolkit (JDT) and compiler that come as part of Eclipse (and which are also used to develop Eclipse itself). However, it can be used for other types of client application as well, see the popular Bit Torrent client for example.

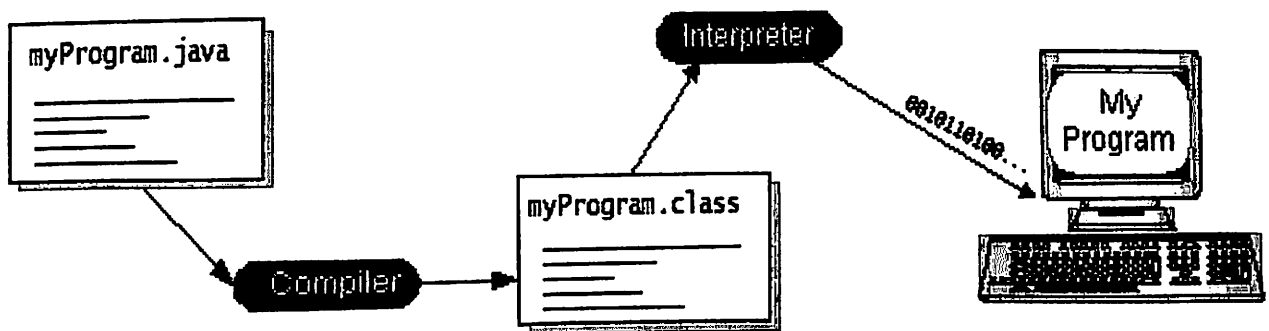
#### **5.3 JAVA**

### 5.3.1 OVERVIEW OF JAVA:

Computers connected to the net are from many different manufacturers, running on different operating systems and they differ in architecture, computing power and capacity. By considering this point SUN

Microsystems Corporation felt the need for a new programming language suitable for this

heterogeneous Environment and java was the solution. This breaks barriers between different computers, chips and operating Systems. .



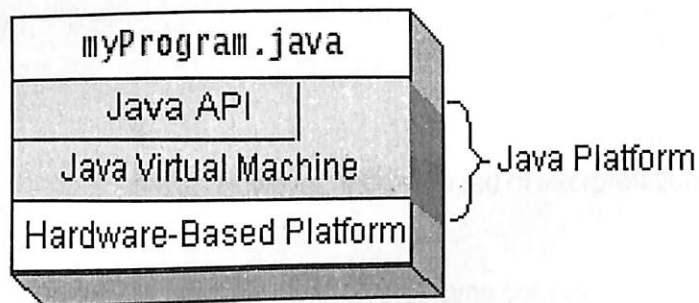
The main properties of the Java, which made Java so popular, are as follows:

- Simple
- Secure
- Portable
- Object-Oriented
- Robust

- Multithreaded
- Interpreted
- High performance

### 5.3.2 The Key Features Of Java Is Byte Code:

The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java runtime systems, which is called the



Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for Byte code. This may come has a bit of surprise.

### 5.3.4 Java Platform:

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. One should be able to write a program once, compile it once, and run it anywhere.

This is achieved by compiling the Java language code, not to machine code but to Java bytecode – instructions analogous to machine code but intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE)

installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host specific features such as graphics, threading and networking. In some JVM versions, bytecode can be compiled to native code, either before or during program execution, resulting in faster execution.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would, and Java suffered a reputation for poor performance. This gap has been narrowed by a number of optimisation techniques introduced in the more recent JVM implementations.

One such technique, known as just-in-time (JIT) compilation, translates Java bytecode into native code the first time that code is executed, then caches it. This result in a program that starts and executes



faster than pure interpreted code can, at the cost of introducing occasional compilation overhead during execution. More sophisticated VMs also use dynamic recompilation, in which the VM analyzes the behavior of the running program and selectively recompiles and optimizes parts of the program.

Dynamic recompilation can achieve optimizations superior to static compilation because the dynamic compiler can base optimizations on knowledge about the runtime environment and the set of loaded classes, and can identify hot spots - parts of the program, often inner loops, that take up the most execution time. JIT compilation and dynamic recompilation allow Java programs to approach the speed of native code without losing portability.

#### **5.4 Android SDK:**

The Android SDK includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include x86-based computers running Linux (any modern desktop Linux Distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. Requirements also include Java Development Kit, Apache Ant, and Python 2.2 or later. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug-in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

A preview release of the Android software development kit (SDK) was released on 12 November 2007.

On 15 July 2008, the Android Developer Challenge Team accidentally sent an email to all entrants in the Android Developer Challenge announcing that a new release of the SDK was available in a "private" download area. The email was intended for winners of the first round of the Android Developer Challenge. The revelation that Google was supplying new SDK releases to some developers and not others (and keeping this arrangement private) have led to widely reported frustration within the Android developer community.

On 18 August 2008 the Android 0.9 SDK beta was released. This release provides an updated and extended API, improved development tools and an updated design for the home screen. Detailed instructions for upgrading are available to those already working with an earlier release. On 23 September 2008 the Android 1.0 SDK (Release 1) was released.<sup>[75]</sup> According to the release notes, it included "mainly bug fixes, although some smaller features were added". It also included several API changes from the 0.9 version.

On March 9, 2009, Google released version 1.1 for the android dev phone. While there are a few aesthetic updates, a few crucial updates include support for "search by voice, priced apps, alarm clock fixes, sending gmail freeze fix, fixes mail notifications and refreshing intervals, and now the maps show business reviews". Another important update is that Dev phones can now access paid apps and

developers can now see them on the Google marketplace.

## Code Snippets :

### 1)Code for Activity :

```
package com.myapp.quiz;

import com.myapp.quiz.Question1Activity.MyCount;
import com.myapp.quiz.R;

import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.graphics.Color;
import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.KeyEvent;
import android.view.View;
import android.widget.ImageView;
import android.widget.CheckBox;
import android.widget.TextView;
import android.view.View.OnClickListener;
import android.widget.Toast;

public class Question1Activity extends Activity implements
View.OnClickListener
{
    DatabaseHandler db = new DatabaseHandler(this);
    CheckBox opt1,opt2,opt3,opt4;

    int givenans;
    private SoundPool sounds1,sounds2,sounds3,sounds4;
    private int buzzer,claps,alarm,beep;
    TextView timeElapsed,text;
    Long highscore=(long) 0;
    int qid=1;
    MyCount counter = new MyCount(21000,1000);

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.question1);

        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    }
}
```

```

highscore= getIntent().getExtras().getLong("highscore");
qid= getIntent().getExtras().getInt("qid");

sounds1 = new SoundPool(10, AudioManager.STREAM_MUSIC,0);
sounds2 = new SoundPool(10, AudioManager.STREAM_MUSIC,0);
sounds3 = new SoundPool(10, AudioManager.STREAM_MUSIC,0);
sounds4 = new SoundPool(10, AudioManager.STREAM_MUSIC,0);
buzzer = sounds1.load(getBaseContext(), R.raw.buzzer, 1);
claps = sounds2.load(getBaseContext(), R.raw.claps, 1);
alarm = sounds3.load(getBaseContext(), R.raw.alarm, 1);
beep = sounds4.load(getBaseContext(), R.raw.beep, 1);

opt1 = (CheckBox) findViewById(R.id.checkBox1);
opt2 = (CheckBox) findViewById(R.id.checkBox2);
opt3 = (CheckBox) findViewById(R.id.checkBox3);
opt4 = (CheckBox) findViewById(R.id.checkBox4);
text = (TextView) findViewById(R.id.timeElapsed);

opt1.setOnClickListener(this);
opt2.setOnClickListener(this);
opt3.setOnClickListener(this);
opt4.setOnClickListener(this);

addListenercheckBox1();
    addListenercheckBox2();
    addListenercheckBox3();
    addListenercheckBox4();
}

public void addListenercheckBox1()
{
    opt1 = (CheckBox) findViewById(R.id.checkBox1);
    opt1.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            switch(v.getId())
            {
                case R.id.checkBox1:
                    givenans=1;
                    break;
                case R.id.checkBox2:
                    givenans=2;
                    break;
                case R.id.checkBox3:
                    givenans=3;
                    break;
                case R.id.checkBox4:
                    givenans=4;
                    break;
            }
            //is chkIos checked?
            if (((CheckBox) v).isChecked() && givenans==3)
            {

```

```

        Toast.makeText(Question1Activity.this,
            "wrong answe re :)", Toast.LENGTH_LONG).show();

        counter.cancel();
        sounds2.play(claps, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
Question2Activity.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 2);
        startActivity(quiz);
        finish();
    }
    else
    {
        counter.cancel();
        sounds1.play(buzzer, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
TryagainActivity1.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 1);
        startActivity(quiz);
        finish();
    }
}
});
}

```

```

public void addListenercheckBox2()

```

```

{

```

```

    opt2 = (CheckBox) findViewById(R.id.checkBox2);

```

```

    opt2.setOnClickListener(new OnClickListener()

```

```

    {

```

```

        @Override

```

```

        public void onClick(View v)

```

```

        {

```

```

            switch(v.getId())

```

```

            {

```

```

                case R.id.checkBox1:

```

```

                    givenans=1;

```

```

                    break;

```

```

                case R.id.checkBox2:

```

```

                    givenans=2;

```

```

                    break;

```

```

                case R.id.checkBox3:

```

```

                    givenans=3;

```

```

                    break;

```

```

                case R.id.checkBox4:

```

```

                    givenans=4;

```

```

                    break;

```

```

            }

```

```

                //is chkIos checked?

```

```

                if (((CheckBox) v).isChecked() && givenans==3)

```

```

                {

```

```

                    Toast.makeText(Question1Activity.this,

```

```

                        "wrong answe re :)", Toast.LENGTH_LONG).show();

```

```

        counter.cancel();
        sounds2.play(claps, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
Question2Activity.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 2);
        startActivity(quiz);
        finish();
    }
    else
    {
        counter.cancel();
        sounds1.play(buzzer, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
TryagainActivity1.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 1);
        startActivity(quiz);
        finish();
    }
}
});
}

public void addListenercheckBox3()
{
    opt3 = (CheckBox) findViewById(R.id.checkBox3);
    opt3.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            switch(v.getId())
            {
                case R.id.checkBox1:
                    givenans=1;
                    break;
                case R.id.checkBox2:
                    givenans=2;
                    break;
                case R.id.checkBox3:
                    givenans=3;
                    break;
                case R.id.checkBox4:
                    givenans=4;
                    break;
            }

            if (((CheckBox) v).isChecked() && givenans==3)
            {
                Toast.makeText(Question1Activity.this,
                    "right answe  :", Toast.LENGTH_LONG).show();

                counter.cancel();
                sounds2.play(claps, 1.0f, 1.0f, 0, 0, 1.0f);
            }
        }
    });
}

```

```

        Intent quiz = new Intent(getApplicationContext(),
Question2Activity.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 2);
        startActivity(quiz);
        finish();
    }
    else
    {
        counter.cancel();
        sounds1.play(buzzer, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
TryagainActivity1.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 1);
        startActivity(quiz);
        finish();
    }
}
});
}

```

```

public void addListenercheckBox4()
{
    opt4 = (CheckBox) findViewById(R.id.checkBox4);
    opt4.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            switch(v.getId())
            {
                case R.id.checkBox1:
                    givenans=1;
                    break;
                case R.id.checkBox2:
                    givenans=2;
                    break;
                case R.id.checkBox3:
                    givenans=3;
                    break;
                case R.id.checkBox4:
                    givenans=4;
                    break;
            }
            if (((CheckBox) v).isChecked() && givenans==3)
            {
                Toast.makeText(Question1Activity.this,
                    "wrong answer :", Toast.LENGTH_LONG).show();

                counter.cancel();
                sounds2.play(claps, 1.0f, 1.0f, 0, 0, 1.0f);
            }
        }
    });
}

```

```

        Intent quiz = new Intent(getApplicationContext(),
Question2Activity.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 2);
        startActivity(quiz);
        finish();
    }
    else
    {
        counter.cancel();
        sounds1.play(buzzer, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
TryagainActivity1.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 1);
        startActivity(quiz);
        finish();
    }
}
});
}

```

```

public boolean onKeyDown(int keyCode, KeyEvent event)
{
    if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() ==
0)
    {
        counter.cancel();
        return super.onKeyDown(keyCode, event);
    }

    return super.onKeyDown(keyCode, event);
}

```

```

@Override
public boolean onKeyUp(int keyCode, KeyEvent event)
{
    if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() ==
0)
    {
        counter.cancel();
        return super.onKeyUp(keyCode, event);
    }

    return super.onKeyUp(keyCode, event);
}

```

```

@Override
protected void onStop()
{
    counter.cancel();
    finish();
    super.onStop();
}

```



```

public class MyCount extends CountdownTimer
{
    public MyCount(long millisInFuture, long countDownInterval)
    {
        super(millisInFuture, countDownInterval);
    }

    @Override
    public void onFinish()
    {
        counter.cancel();
        sounds3.play(alarm, 1.0f, 1.0f, 0, 0, 1.0f);
        Intent quiz = new Intent(getApplicationContext(),
AlarmActivity1.class);
        quiz.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        quiz.putExtra("highscore", highscore);
        quiz.putExtra("qid", 1);
        startActivity(quiz);
        finish();
    }

    @Override
    public void onTick(long millisUntilFinished)
    {
        Long sec=millisUntilFinished/1000;
        text.setText(Long.toString(millisUntilFinished/1000));
        Long time=(long) 21000/1000;
        highscore=highscore + (time-sec);
        if(sec<=5)
        {
            sounds4.play(beep, 1.0f, 1.0f, 0, 0, 1.0f);
            text.setTextColor(Color.parseColor("#FF0000"));
        }
    }
}

@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
}
}

```

## 2-)Code for layout

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/img"
    android:orientation="vertical" >

```

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="442dp"
    android:orientation="vertical"
    android:padding="10dip" >
```

```
<TextView
    android:id="@+id/question1text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="10dip"
    android:text="Question1 -What is the name of the flashing
symbol on the computer screen that shows where the information you enter
will appear?"
    android:textColor="#000000"
    android:textSize="20dip"
    android:textStyle="bold" />
```

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="295dp"
    android:layout_height="100dp"
    android:src="@drawable/motu" />
```

```
<CheckBox
    android:id="@+id/checkbox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="false"
    android:text="Chip"
    android:textColor="#000000"
    android:textStyle="bold" />
```

```
<CheckBox
    android:id="@+id/checkbox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="false"
    android:text="Circuit"
    android:textColor="#000000"
    android:textStyle="bold" />
```

```
<CheckBox
    android:id="@+id/checkbox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="false"
    android:text="Cursor"
    android:textColor="#000000"
    android:textStyle="bold" />
```

```
<CheckBox
    android:id="@+id/checkbox4"
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"  
android:text="Bit"  
  android:textColor="#000000"  
  android:textStyle="bold" />
```

```
</LinearLayout>
```

```
</ScrollView>
```

## CHAPTER 6

## TESTING

### 6.1 What is testing?

A process of executing a program with the explicit intention of finding errors, that is making the program fail. Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

### 6.2 Psychology of Testing

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing if the intent of showing that a program works but the intent should be to show that a program does not work. Testing is the process of executing a program with the intent of finding errors.

### 6.3 Testing Objectives

- The main objective of testing is to uncover a host of errors, systematically and with minimum

effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet discovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

#### **6.4 Software Testing:**

It is the process of testing the functionality and correctness of a software by running it. A good test case is the one that has a high probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error. Software testing is usually performed for one of the two reasons.

- Defect detection.
- Reliability estimation.

#### **6.5 Black Box Testing:**

Black box testing is based on the software's specifications or requirements, without reference to its internal workings

Black Box Testing is not a type of testing; it instead is a testing strategy, which does not need any knowledge of internal design or code etc. As the name "black box" suggests, no knowledge of internal logic or code structure is required. The types of testing under this strategy are totally based/focused on the testing for requirements and functionality of the work product/software application. Black box testing is sometimes also called as "Opaque Testing", "Functional/Behavioral Testing" and "Closed Box Testing".

The base of the Black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming common to route the Testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test the application by the developer.

In order to implement Black Box Testing Strategy, the tester is needed to be thorough with the requirement specifications of the system and as a user, should know, how the system should behave in response to the particular action.

## **6.6 White Box Testing:**

White box testing is a security testing method that can be used to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities. White box testing is performed based on the knowledge of how the system is implemented. White box testing includes analyzing data flow, control flow, information flow, coding practices, and exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

White box testing requires access to the source code. Though white box testing can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing during the unit testing phase.

White box testing requires knowing what makes software secure or insecure, how to think like an attacker, and how to use different testing tools and techniques. The first step in white box testing is to comprehend and analyze source code, so knowing what makes software secure is a fundamental requirement. Second, to create tests that exploit software, a tester must think like an attacker. Third, to perform testing effectively, testers need to know the different tools and techniques available for white box testing. The three requirements do not work in isolation, but together.

Knowing the internal working i.e., to test if all internal operations are performed according to program structures and data structures. To test if all internal components have been adequately exercised.

## **6.7 Levels of Testing:**

In order to uncover the errors present in different phases we have the concept of levels of testing. The

basic levels of testing are

Client needs	Acceptance Testing
Requirements	System Testing
Design	Integration Testing
Code	Unit Testing

## **6.8 Software Testing Strategies:**

A strategy for software testing will begin in the following order.

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

### **6.8.1 Unit Testing:**

It concentrates on each unit of the software as implemented in source code and is a white box oriented. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. In the unit testing, the steps can be conducted in parallel for multiple components. In my project I tested all the modules individually related to main function codes and attacks also.

### **6.8.2 Integration Testing:**

Here focus is on design and construction of the software architecture. Integration Testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules.

This testing activity can be considered as testing the design and hence the emphasis on testing module interactions. In this project the main system is formed by integrating all the modules. When integrating all the modules I have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before integration.



### **6.8.3 Validation Testing:**

In this, requirements established as part of software requirement analysis are validated against the software that has been constructed i.e., validation succeeds when software functions in a manner that can reasonably expected by the customer.

### **6.8.4 System Testing:**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements.

### **6.9 TEST CASES:**

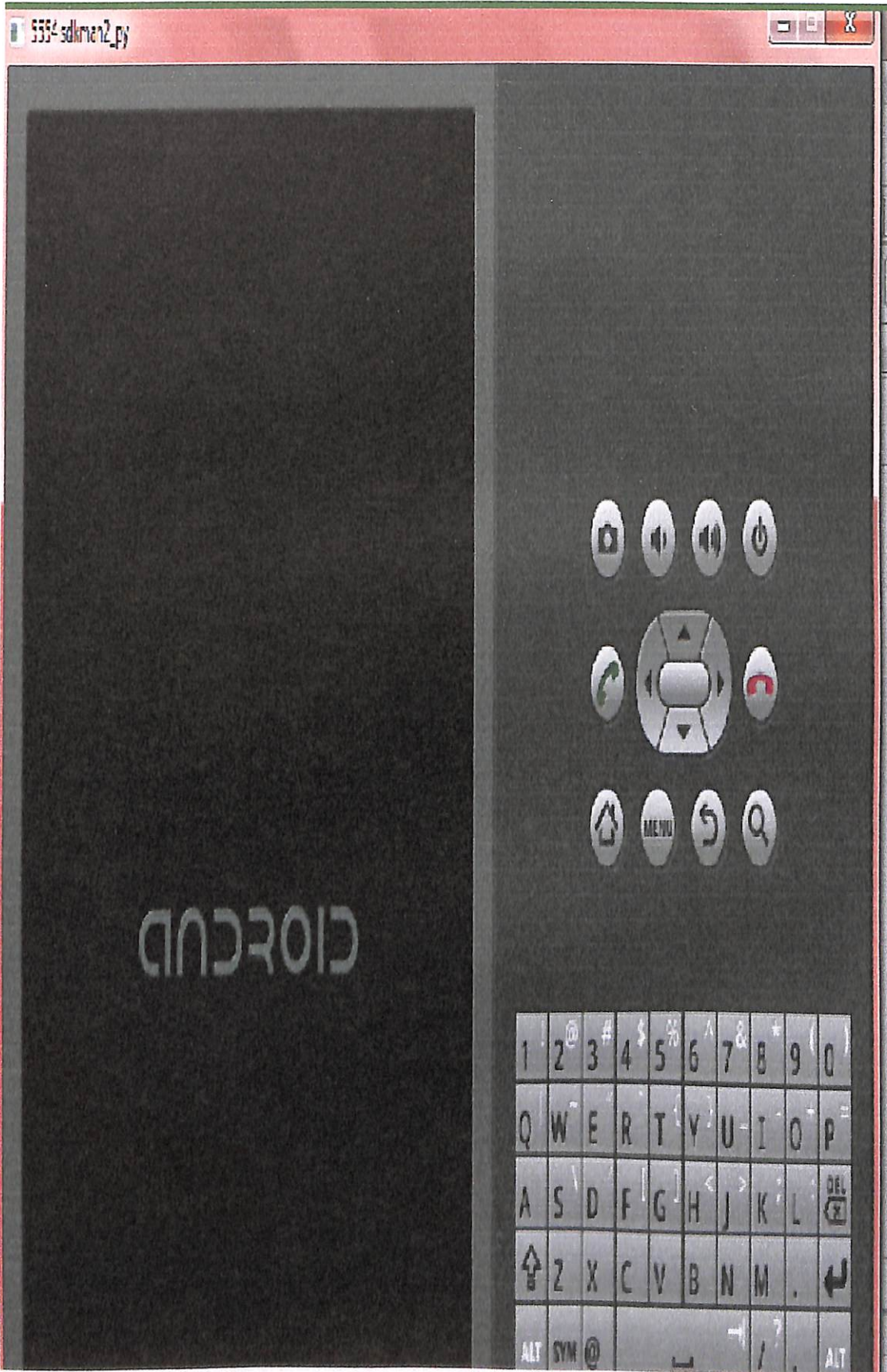
A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system meets specifications. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle.

In some settings an oracle could be a requirement or use case. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to

as test scripts, particularly when written. Written test cases are usually collected into test suites.

### **6.9.1 What are positive and negative test cases?**

- A positive test case is when the test is designed to return what is expected according to the requirement.
  
- Negative test case is when the test is designed to determine the response of the product outside of what is defined.





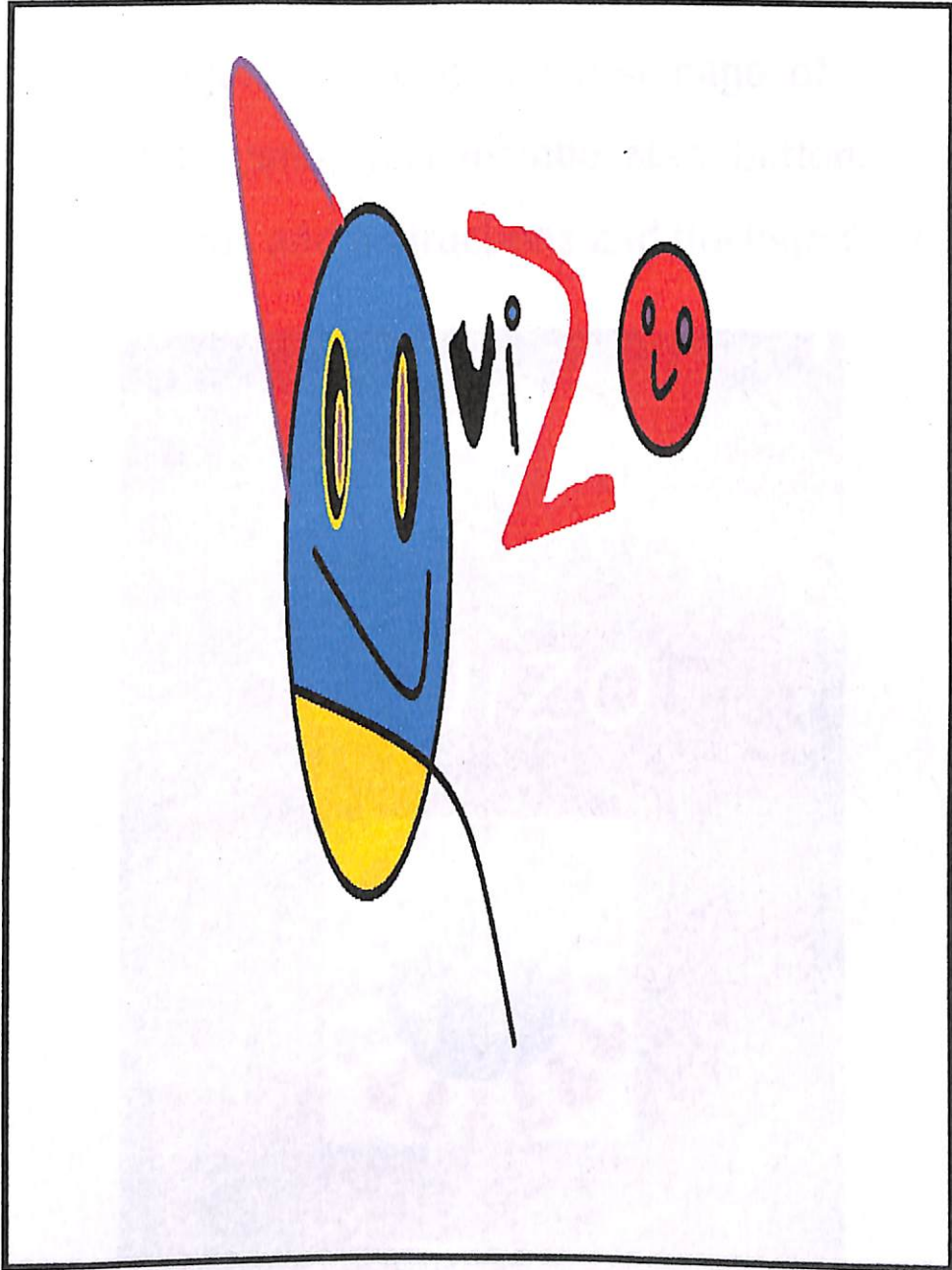


**Figure 1**

**In Figure 1, The Android Main screen displaying “Quizo” icon.**

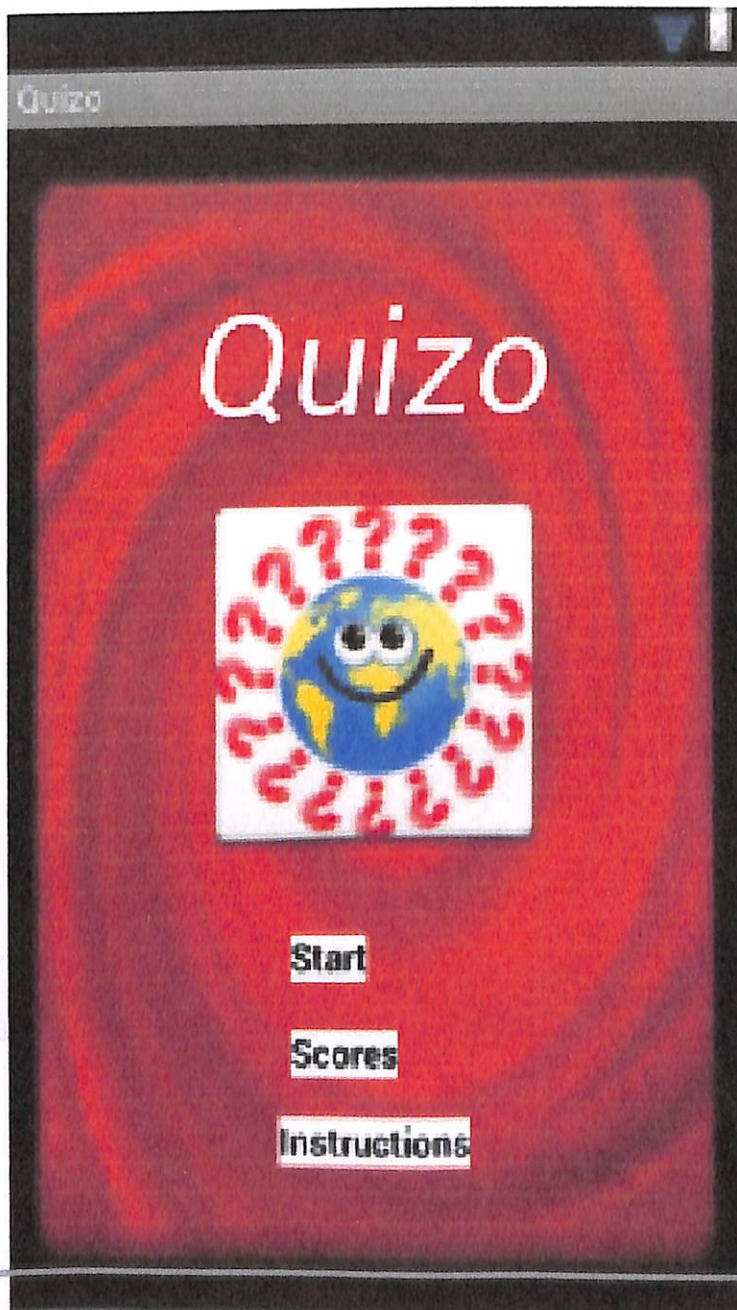
## Figure 2

Figure 2 shows the “Quizo” icon.



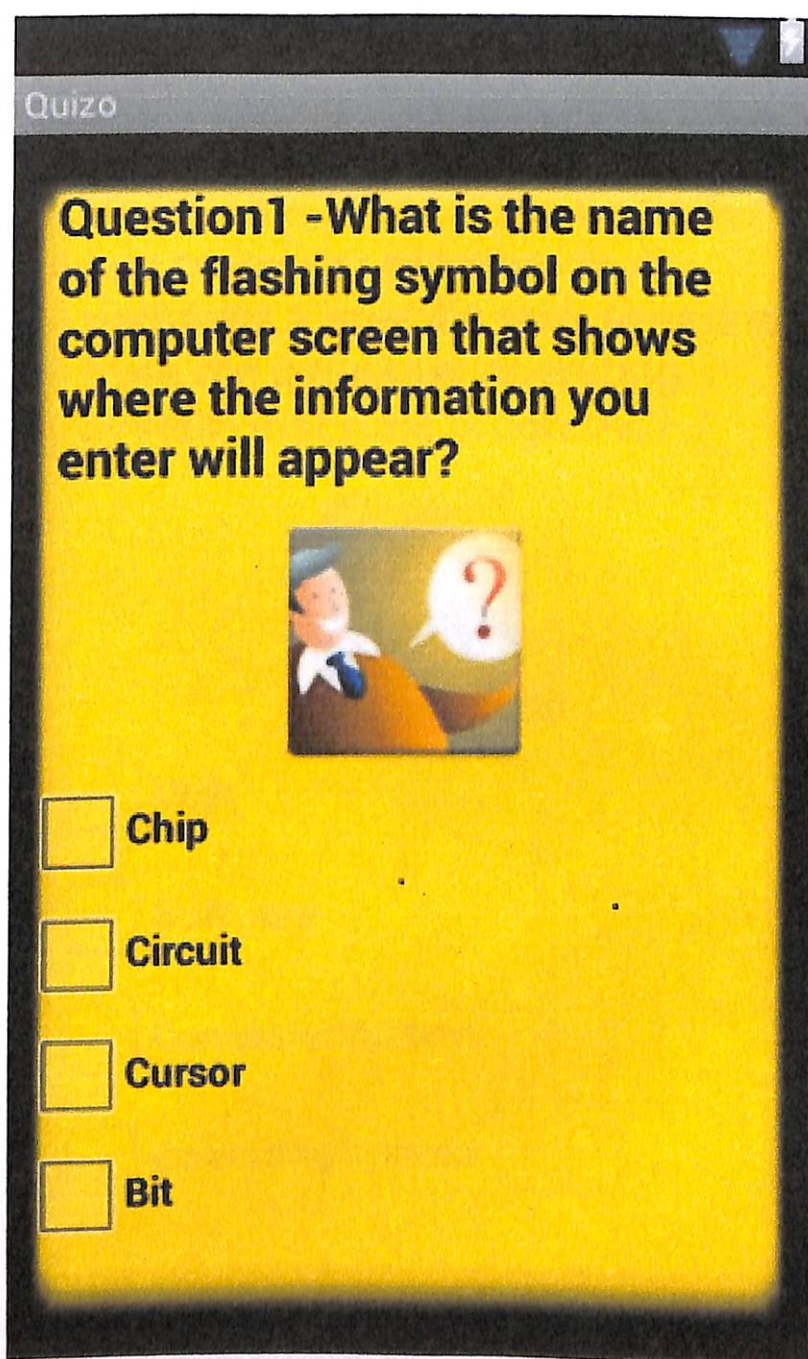
## Figure 3

Figure 3 shows the first page of the game which include start button, scores and instructions and the logo Quizo.



## Figure 4

Includes multiple choice questions related to computers.






## Figure 5

Includes multiple choice questions related to computers.

Quizo

**Question2- What is the name for any computer program that performs a specific task separate from the computer operating system that runs programs?**



**Disk**

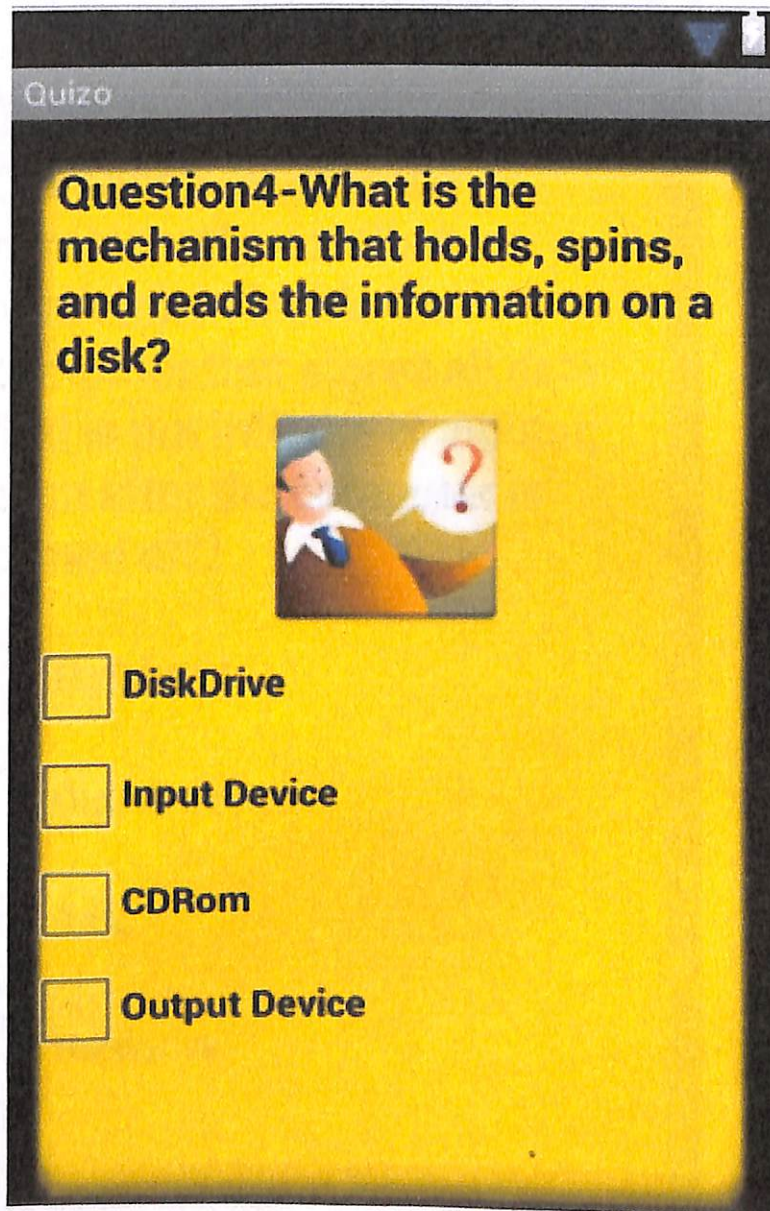
**Software**

**ComputerSystem**

**OperatingSystem**

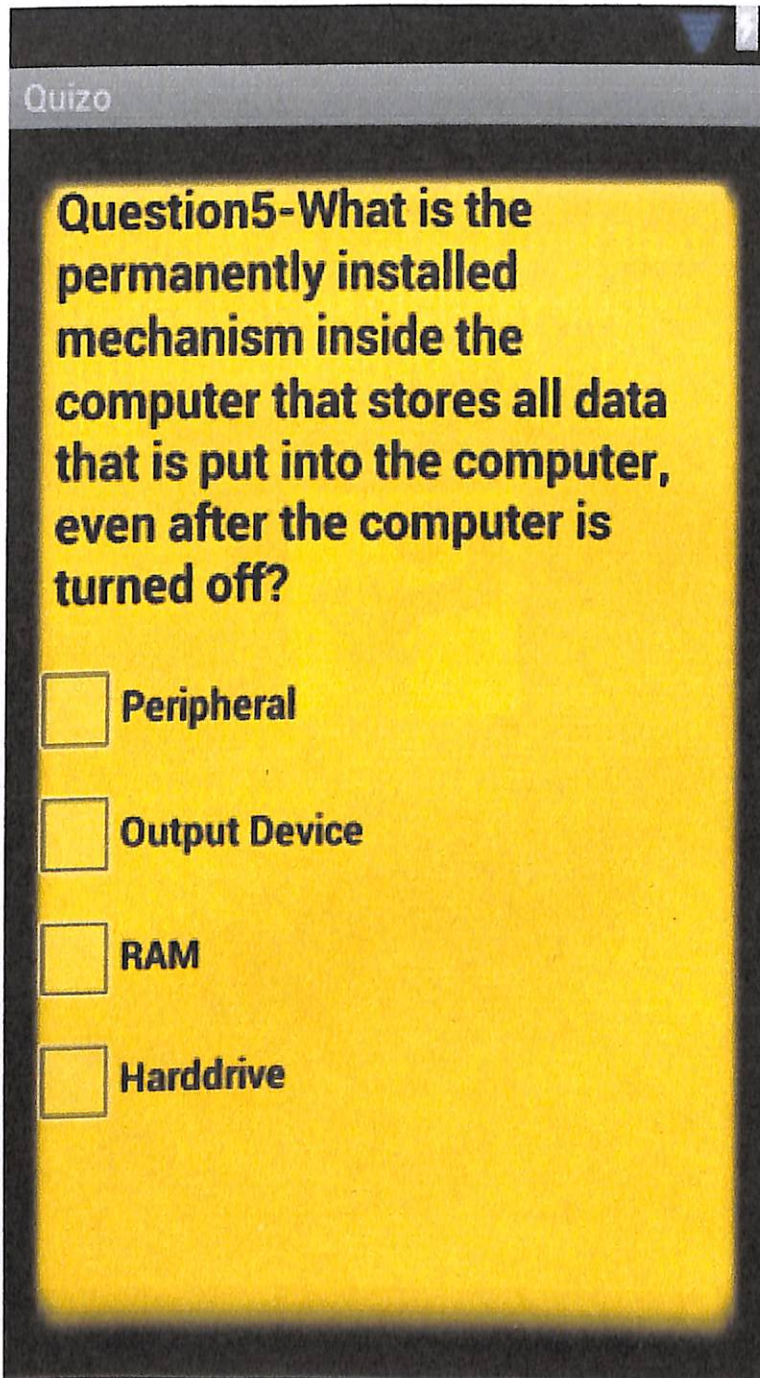
## Figure 6

Includes multiple choice questions related to computers.



## Figure 7

Includes multiple choice questions related to computers.



The image shows a screenshot of a quiz application. At the top, there is a dark grey header with the word "Quizo" in white. Below the header, the question is displayed in bold black text on a yellow background. The question asks for the permanently installed mechanism inside a computer that stores all data, even after the computer is turned off. There are four multiple-choice options, each with a square checkbox to its left. The options are: Peripheral, Output Device, RAM, and Harddrive.

Quizo

**Question5-What is the permanently installed mechanism inside the computer that stores all data that is put into the computer, even after the computer is turned off?**

Peripheral

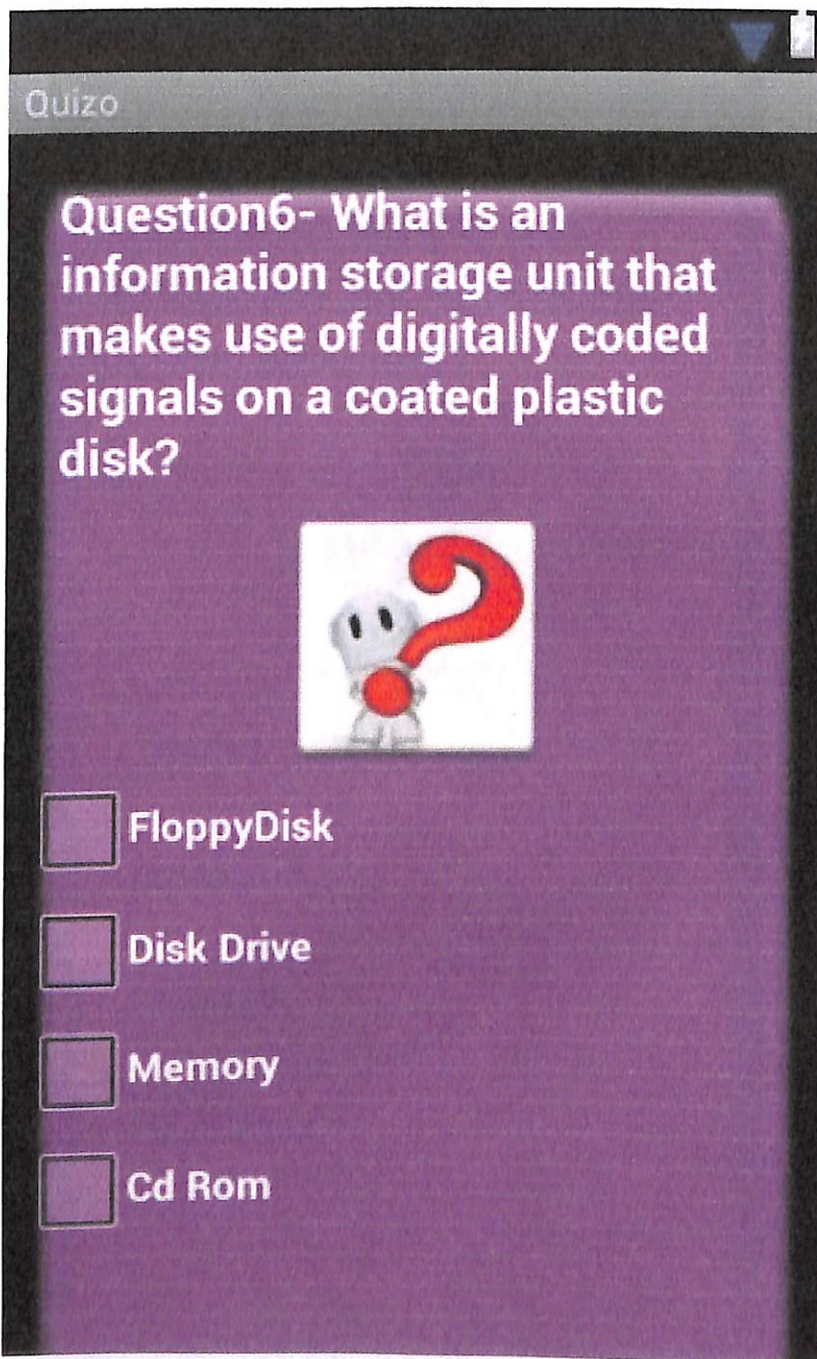
Output Device

RAM

Harddrive

## Figure 8

Includes multiple choice  
questions related to computers




## Figure 9

Includes multiple choice questions related to computers

Quizo

**Question7-What is the primary input device used to enter information and instructions into the computer?**



Mouse

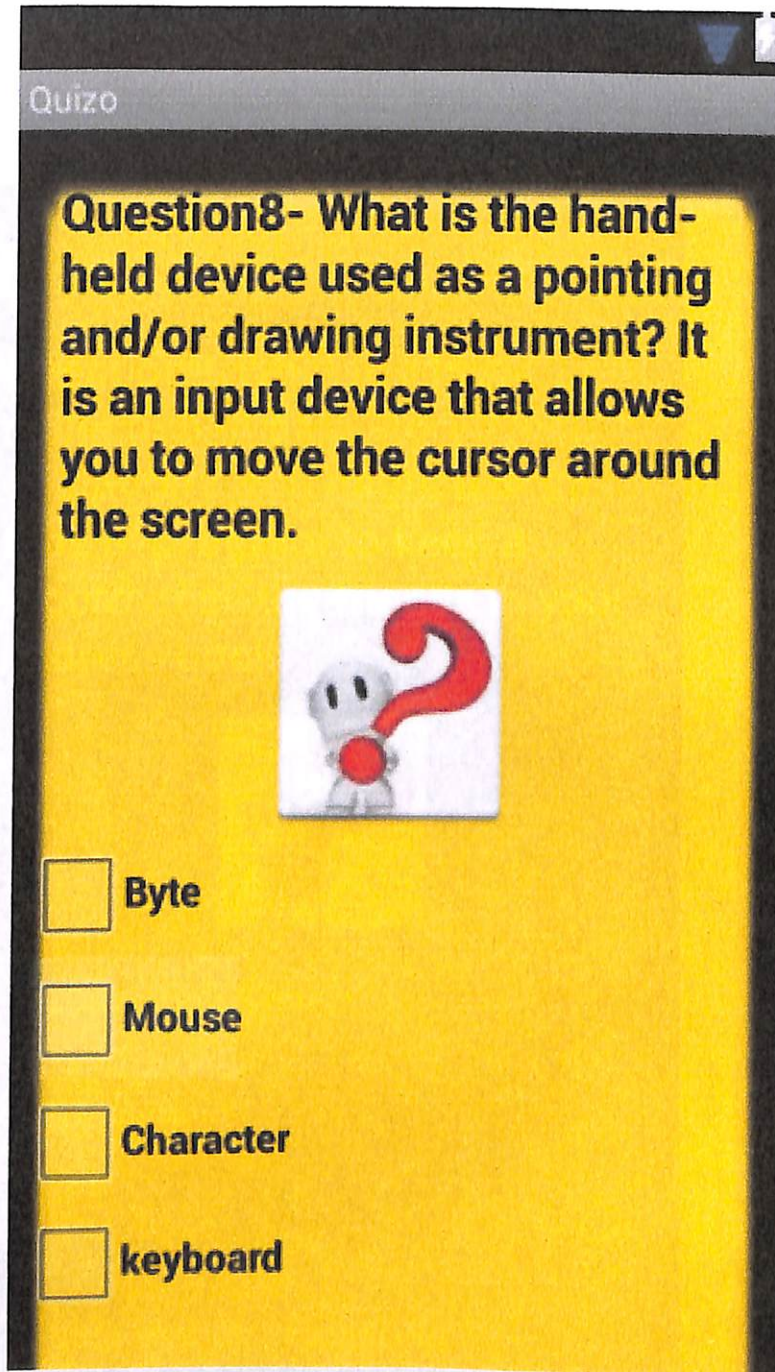
Keyboard

Scanner

Disk Drive

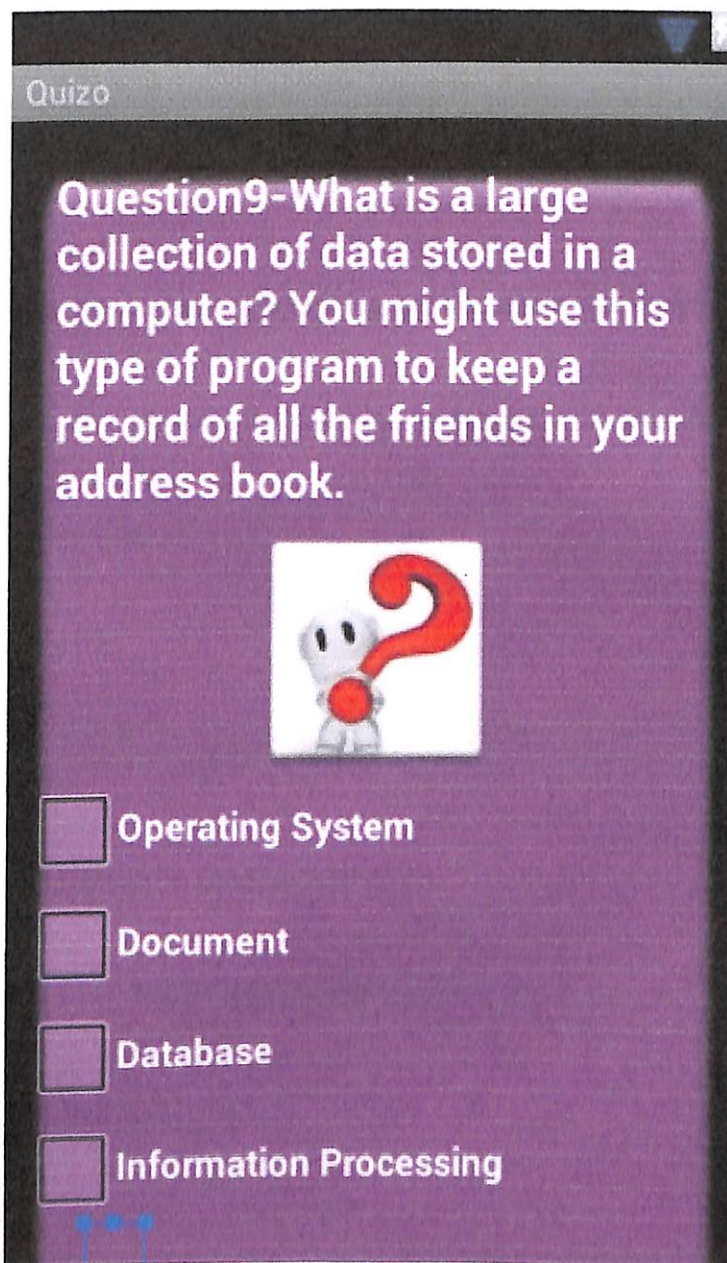
## Figure 10

Includes multiple choice questions related to computers



## Figure 11

Includes multiple choice questions related to computers



## **CHAPTER 8**

## **FUTURE SCOPE**

Currently the application only contains the technical questions without any level, the project can also be extended to have different levels with questions related to different domain.

## **CHAPTER 9**

## **CONCLUSION**

Finally we concluded that the Quizo is an android game application developed on a new mobile platform Android, and for the users who loves playing games, it also helps users to check their knowledge related to computers.

## **CHAPTER 10**

## **BIBLIOGRAPHY**

[1] Android Application Development ,

[2] Android & its application, <http://www.android.com/>

## **CHAPTER 11**

## **APPENDIX**



The following terms are used in these documents.

## **1..apk extension**

The extension for an Android package file, which typically contains all of the files related to a single Android application. The file itself is a compressed collection of an AndroidManifest.xml file, application code (.dex files), resource files, and other files. A project is compiled into a single .apk file.

## **2.dex extension**

Android programs are compiled into .dex (Dalvik Executable) files, which are in turn zipped into a single .apk file on the device. .dex files can be created by automatically translating compiled applications written in the Java programming language.

## **3.Action**

A description of something that an Intent sender wants done. An action is a string value assigned to Intent. Action strings can be defined by Android or by a third-party developer. For example, android.intent.action.VIEW for a Web URL, or com.example.rumbler.SHAKE\_PHONE for a custom application to vibrate the phone.

## **4.Activity**

A single screen in an application, with supporting Java code, derived from the Activity class.

## **5.Adb**

Android Debug Bridge, a command-line debugging application shipped with the SDK. It provides tools to browse the device, copy tools on the device, and forward ports for debugging. See Using adb for more information.

## **6.Application**

A collection of one or more activities, services, listeners, and intent receivers. An application has a single manifest, and is compiled into a single .apk file on the device.

## **7.Content Provider**

A class built on ContentProvider that handles content query strings of a specific format to return data in a specific format. See Reading and writing data to a content provider for information on using content providers.

## **8.Content URI**

A type of URI. See the URI entry.

## **9.Dalvik**

The name of Android's virtual machine. The Dalvik VM is an interpreter-only virtual machine that executes files in the Dalvik Executable (.dex) format, a format that is optimized for efficient storage and memory-mappable execution. The virtual machine is register-based, and it can run classes compiled by a Java language compiler that have been transformed into its native format using the included "dx" tool.

The VM runs on top of Posix-compliant operating systems, which it relies on for underlying functionality (such as threading and low level memory management). The Dalvik core class library is intended to provide a familiar development base for those used to programming with Java Standard Edition, but it is geared specifically to the needs of a small mobile device.

## **10.DDMS**

Dalvik Debug Monitor Service, a GUI debugging application shipped with the SDK. It provides screen capture, log dump, and process examination capabilities. See [Using the Dalvik Debug Monitor Server](#) to learn more about this program.

## **11.Drawable**

A compiled visual resource that can be used as a background, title, or other part of the screen. It is compiled into an `android.graphics.drawable` subclass.

## **12.Intent**

A class (Intent) that contains several fields describing what a caller would like to do. The caller sends this intent to Android's intent resolver, which looks through the intent filters of all applications to find the activity most suited to handle this intent. Intent fields include the desired action, a category, a data string, the MIME type of the data, a handling class, and other restrictions.

## **13.Intent Filter**

Activities and intent receivers include one or more filters in their manifest to describe what kinds of intents or messages they can handle or want to receive. An intent filter lists a set of requirements, such as data type, action requested, and URI format, that the Intent or message must fulfill. For activities, Android searches for the activity with the most closely matching valid match between the Intent and the activity filter. For messages, Android will forward a message to all receivers with matching intent filters.

## **14.Intent Receiver**

An application class that listens for messages broadcast by calling `Context.sendBroadcast ()`. For example code, see [Listening for and broadcasting global messages](#).

## **15.Layout resource**

An XML file that describes the layout of an Activity screen.

## **16.Manifest**

An XML file associated with each Application that describes the various activities, intent filters, services, and other items that it exposes. See [AndroidManifest.xml File Details](#).

## **17. URIs**

Android uses URI strings both for requesting data (e.g., a list of contacts) and for requesting actions (e.g., opening a Web page in a browser). Both are valid URI strings, but have different values. All requests for data must start with the string "content://". Action strings are valid URIs that can be handled appropriately by applications on the device; for example, a URI starting with "http://" will be handled by the browser.