A Project on

**CALENDROID**

Submitted in partial fulfillment of award of

**BACHELOR OF TECHNOLOGY**

Degree

In

**INFORMATION TECHNOLOGY**

By

Ritvik Khattar 12615603109

Piyush Kalra 14015603109

Inderjeet Khurana 16015603109

Under the supervision of: HOD (IT)

**Assistant Prof. Neha Sharma Dr. Shafiq**



**Northern India Engineering College,**

**Approved by AICTE, Affiliated to GGSIPU, Shastri Park**

**Near Kashmere Gate ISBT, Delhi**

**May,2013**

**ACKNOWLEDGEMENT**

Without the moral support and guidance from a number of people, this project would never have been complete. We wish to express our gratitude to those people who have directly or indirectly contributed towards successful completion of this project.

We would like to thank Prof. Neha Sharma, for her valuable suggestions, support and encouragement. We would also like to express our gratitude to her for making special efforts to study and review our project and give us feedback.

**CONTENTS**

**DECLARATION……………………………………………………………………..09**

**ABSTRACT…………………………………………………………………………..10**

**1. INTRODUCTION………………………………………………………………....11**

1.1. Android…..……………………………………………………………………..11

1.1.1. Specification……...……………………………………………………...13

1.1.2. Characteristics……………..…………………………………………….14

1.2. Eclipse……………………….………………………………………………….14

1.2.1. ADT Plugin………………....…………………………………………...15

1.3. Android Virtual Device.………………………………………………………..15

**2. FEASIBILITY……………………………………………………………………..17**

2.1. Feasibility Study Process……………………………………………………….17

2.2. Importance of Feasibility Analysis……………………………………………..17

2.3. Feasibility Types………………………………………………………………..17

2.3.1. Economic Feasibility…………………………………………………….18

2.3.2. Technical Feasibility………………………………………………....….18

2.3.3. Social Feasibility………………………………………………………...18

2.3.4. Legal Feasibility………………………………………………………....19

2.3.5. Operational Feasibility…………………………………………………..19

**3. SOFTWARE REQUIREMENT SPECIFICATION……………………………20**

3.1 Introduction……………………………………………………………………20

3.1.1 Purpose…………………………………………………………………20

3.1.2 Scope…………………………………………………………………...20

3.1.3 Definitions…………………..………………………………………….20

3.1.4 Overview……………………………………………………………….20

3.2 Overall Description……………………………………………………….……20

3.2.1 Product Perspective…………………………………………….….……21

3.2.2 Product Functions…………………………………………….………...21

3.2.3 User Characteristics………………………………………………….….22

3.2.4 Constraints…………………………………………………………..…..22

3.2.5 Assumptions and Dependencies…………………………………….…..22

3.2.6 Apportioning of Requirements…………………………………….……22

3.3 Specific Requirements…………………………………………………….……22

3.3.1 Functional Requirements……………………………………….……….22

3.3.2 Performance Requirements……………………………………….……..23

3.3.3 Non-Functional Requirements…………………………………….…….23

3.4 Software System Attributes……………………………………………………24

3.4.1 Security…………………………………………………………………..24

3.4.2 Maintainability…………………………………………………………...24

3.4.3 Portability………………………………………………………………..24

**4. DESIGN PHASE…………………………………………………………….……25**

4.1 Use Case Diagram………………………………………………………….…..25

4.1.1 What is Use Case…………………………………………………….….25

4.1.2 What Does a Use Case Describe…………………………………….….25

4.1.3 Characteristics…………………………………………………….…….25

4.1.4 How to Write Use Case……………………………………………..…..26

4.1.5 Use Case for Calendroid…...…………………………………………....27

4.2 Data Flow Diagrams………………...…………………………………………..28

4.2.1 What are DFDs………………...………………………………………...28

4.2.2 How to Draw DFDs……………..………………………………………28

4.2.3 0-Level DFD……………………..……………………………………....29

4.2.4 1-Level DFD………………………..……………………………………29

4.2.5 DFD for Calendroid…...……………..…………………………………..30

4.3. Entity Relationship Diagram…...……………………………………………….32

4.3.1. What is E-R Diagram……..……………………………………….…….32

4.3.2. E-R Notations……………..………………………………………..……32

4.3.3. E-R Diagram for Calendroid…....………………………………….……35

**5. IMPLEMENTATION……………………………………………………….…....36**

5.1. Technology used……………………………………………………………….36

5.1.1. About Java……………………………………………………………...36

5.2. Classes Used…………………………………………………………………...37

5.2.1. Database.java……………………………………………………………37

5.2.2. DisplayMenu.java………………………………………………………..37

5.2.3. Photos.java……………………………………………………………….38

5.2.4. Reader.java…….…………………………………………….…………38

5.2.5. SimpleCalendarViewActivity.java………………………..……………39

5.2.6. Videos.java……………………………………………………………...40

5.2.7. Writer.java………..……………………………………………………..40

**6. TESTING……………………………………………………………………….….42**

6.1 Artefacts of Testing………………………………………………………….…42

6.2. Types of Testing…………………………………………………………….….43

6.2.1. Performance Testing………………………………………………….…43

6.2.2. Security Testing…………………………………………………………44

6.2.3. Exploratory Testing……………………………………………………..45

6.2.4. Functional Testing………………………………………………………45

6.2.5. User Interface Testing…………………………………………………..46

6.2.6. Volume Testing…………………………………………………………46

6.2.7. Stress Testing…………………………………………………………...46

6.2.8. Load Testing……………………………………………………………46

6.2.9. Installation Testing……………………………………………………..46

6.2.10. Configuration Testing…………………………………………………46

6.2.11. Compatibility Testing…………………………………………………46

6.2.12. Documentation testing………………………………………….……..47

6.2.13 Error testing…………………………………………………….….…...47

6.2.14 Comparison Testing…………………………………………….……...47

6.2.15 Acceptance Testing……………………………………………….……47

6.2.16 Alpha Testing………………………………………………….……….47

6.2.17 Beta Testing……………………………………………………………47

6.2.18 Validation Testing……………………………………………….……..47

6.2.19 Output Testing…………………………………………………………48

6.2.20 Regression Testing…………………………………………….……….48

6.2.21 Incremental Integration Testing………………………………….…….48

6.2.22 Usability Testing……………………………………………….………48

6.2.23 Unit Testing…………………………………………………………....48

6.2.24 Integration Testing…………………………………………………….48

6.2.25 System Testing…………………………………………………….…..49

6.2.26 Parallel Testing…………………………………………………….…..49

**7. SCOPE OF WORK………………………………………………………………50**

7.1 Scope……………………………………………………………………….…..50

**8. DISCUSSION AND RESULT…………………………………………………..52**

8.1 Introduction……………………………………………………………………52

8.2 JDK……………………………………………………………………….……52

8.3 Eclipse……………………………………………………………………….…53

8.4 Result……………………………………………………………………….….54

**LIST OF FIGURES**

4.1. Use Case Diagram……………………………………………………………….27

4.2. Data Flow Diagram……………………………………………………………...30

4.3. E-R Diagram…………………………………………………………………….35

5.1. Display Menu Window………………………………………………………….37

5.2. Photos Window..……………………………..…………………………………38

5.3. Reader Window.……..………………………………………………………….39

5.4. Calendar Window……………………………………………………………….39

5.5. Videos Window…………………………………………………………………40

5.6. Writer Window...………………………………………………………………..41

**DECLARATION**

This is to certify that the project entitled ‘Calendroid’, submitted by Smeechi Mittal, Piyush Kalra and Inderjeet Khurana in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Information Technology Engineering from Northern India Engineering College, Guru Gobind Singh Indraprastha University, Delhi, comprises only our original work and due acknowledgement has been made in the text to all other work or material used.

**Name of Students:**  **Name of mentor:**

Smeechi Mittal Assistant Prof. Neha Sharma

Piyush Kalra

Inderjeet Khurana

**ABSTRACT**

Calendroid is an application developed in Android environment. Android is a [Linux](http://en.wikipedia.org/wiki/Linux)-based [operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) designed primarily for [touchscreen](http://en.wikipedia.org/wiki/Touchscreen) mobile devices such as [smartphones](http://en.wikipedia.org/wiki/Smartphone) and [tablet computers](http://en.wikipedia.org/wiki/Tablet_computer). Android is [open source](http://en.wikipedia.org/wiki/Open_source) and Google releases the code under the [Apache License](http://en.wikipedia.org/wiki/Apache_License). This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Applications are developed in the [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) language using the [Android software development kit](http://en.wikipedia.org/wiki/Android_SDK) (SDK). The SDK includes a comprehensive set of development tools,[[1]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-54) including a [debugger](http://en.wikipedia.org/wiki/Debugger), [software libraries](http://en.wikipedia.org/wiki/Software_library), a handset [emulator](http://en.wikipedia.org/wiki/Emulator) based on [QEMU](http://en.wikipedia.org/wiki/QEMU), documentation, sample code, and tutorials. The officially supported [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is [Eclipse](http://en.wikipedia.org/wiki/Eclipse_%28software%29) using the Android Development Tools (ADT) plugin. Other development tools are available, including a [Native Development Kit](http://en.wikipedia.org/wiki/Android_NDK) for applications or extensions in C or C++, [Google App Inventor](http://en.wikipedia.org/wiki/Google_App_Inventor), a visual environment for novice programmers, and various [cross platform mobile web applications frameworks](http://en.wikipedia.org/wiki/Multiple_phone_web_based_application_framework).[1]

In [computer programming](http://en.wikipedia.org/wiki/Computer_programming), Eclipse is a multi-language [software development environment](http://en.wikipedia.org/wiki/Software_development_environment) comprising a base [workspace](http://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_%28computing%29) system for customizing the environment. It is written mostly in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29). It can be used to develop applications in Java and, by means of various plug-ins, other [programming languages](http://en.wikipedia.org/wiki/Programming_language) including [Ada](http://en.wikipedia.org/wiki/Ada_%28programming_language%29), [C](http://en.wikipedia.org/wiki/C_%28programming_language%29), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [COBOL](http://en.wikipedia.org/wiki/COBOL), [Fortran](http://en.wikipedia.org/wiki/Fortran), [Haskell](http://en.wikipedia.org/wiki/Haskell_%28programming_language%29), [JavaScript](http://en.wikipedia.org/wiki/JavaScript), [Perl](http://en.wikipedia.org/wiki/Perl), [PHP](http://en.wikipedia.org/wiki/PHP), [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29), [R](http://en.wikipedia.org/wiki/R_%28programming_language%29), [Ruby](http://en.wikipedia.org/wiki/Ruby_%28programming_language%29) (including [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) framework), [Scala](http://en.wikipedia.org/wiki/Scala_%28programming_language%29), [Clojure](http://en.wikipedia.org/wiki/Clojure), [Groovy](http://en.wikipedia.org/wiki/Groovy_%28programming_language%29), [Scheme](http://en.wikipedia.org/wiki/Scheme_%28programming_language%29), and [Erlang](http://en.wikipedia.org/wiki/Erlang_%28programming_language%29). It can also be used to develop packages for the software [Mathematica](http://en.wikipedia.org/wiki/Mathematica). Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application.

The project made by us, called Calendroid is developed using Eclipse Indigo version 2.0 and ADT plugin. It can be used by anyone who has an Android based phone with API level of 10 or above. The application appears as an icon on the android phone’s screen. The icon has an image of the green coloured android mascot. On clicking the icon, a calendar window opens. The window shows the calendar dates of the current month. User can switch between the months to view any of the year’s/month’s date. The user can save data, photographs or videos corresponding to any date in the calendar.

**Chapter 1**

**INTRODUCTION**

Calendroid is an Android based application developed using Java in Eclipse, with the help of Android Development Tools plugin. The application can be used by smartphone that has Android operating system with API level of 10 or above. The application is visible on phone’s main screen as an icon with the image of android. If the user wants to access the application, he can click on the icon image. A window appears that shows current month’s calendar. The user can switch between different months or years to access any month’s/year’s date schedule. All the dates of the calendar are clickable. Once the user clicks on any of the dates, the screen comes up with three different options : journal, camera and video.

If the user clicks on journal, a writing pad appears. User can type any desired text/ reminders/ notes or any data of his requirement. The data gets saved in an automatically created file simultaneously. The file is named after the calendar date in which it is typed. Later, if the user re-opens the same date’s journal, then the screen would show the text that the user had typed earlier. Also, the user is free to make changes or add any further desired information corresponding to same date.

Similarly if the user opts for camera, the front camera of the cell phone opens and the user can click photographs. These photographs get saved in the chosen date of the calendar. User can access the photos by looking into the same date of calendar. On clicking the video option, camera opens in the video mode. User can shoot videos which get saved corresponding to the selected date of the calendar. This video can be accessed by revisiting the same date of the calendar.[2]

Users are free to choose any date from the calendar to store data. But, it is not possible to add back dated text, photographs or videos. The date which is older than the present date can be opened but only in read mode, it is not possible to write anything in previously dated files. Thus, this enables user to set reminders or save data for future dates only, preventing any tampering of previous data.

**1.1. ANDROID**

Android consists of a [kernel](http://en.wikipedia.org/wiki/Kernel_%28computing%29) based on [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel) version 2.6 and, from Android 4.0 Ice Cream Sandwich onwards, version 3.x, with [middleware](http://en.wikipedia.org/wiki/Middleware), [libraries](http://en.wikipedia.org/wiki/Software_library) and [APIs](http://en.wikipedia.org/wiki/Application_programming_interface) written in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29), and [application software](http://en.wikipedia.org/wiki/Application_software) running on an [application framework](http://en.wikipedia.org/wiki/Application_framework) which includes Java-compatible libraries based on [Apache Harmony](http://en.wikipedia.org/wiki/Apache_Harmony). Android uses the [Dalvik virtual machine](http://en.wikipedia.org/wiki/Dalvik_%28software%29) with [just-in-time compilation](http://en.wikipedia.org/wiki/Just-in-time_compilation) to run Dalvik 'dex-code' (Dalvik Executable), which is usually translated from [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode).[[58]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-58) The main hardware platform for Android is the [ARM architecture](http://en.wikipedia.org/wiki/ARM_architecture). There is support for [x86](http://en.wikipedia.org/wiki/X86) from the [Android x86](http://en.wikipedia.org/wiki/Android_x86) project,[[9]](http://en.wikipedia.org/wiki/Android_%28operating_system%29" \l "cite_note-ARMAN-4.0-on-x86-9) and [Google TV](http://en.wikipedia.org/wiki/Google_TV) uses a special x86 version of Android. Since 2008, Android has seen [numerous updates](http://en.wikipedia.org/wiki/Android_version_history) which have incrementally improved the operating system, adding new features and fixing bugs in previous releases. Each major release is named in alphabetical order after a dessert or sugary treat; for example, version 1.5 Cupcake was followed by 1.6 Donut.

Android's user interface is based on [direct manipulation](http://en.wikipedia.org/wiki/Direct_manipulation_interface), using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching and reverse pinching to manipulate on-screen objects. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide [haptic feedback](http://en.wikipedia.org/wiki/Haptic_feedback) to the user. Android devices boot to the homescreen, the primary navigation and information point on the device, which is similar to the [desktop](http://en.wikipedia.org/wiki/Desktop_metaphor) found on PCs. Android homescreens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content such as the weather forecast, the user's email inbox, or a [news ticker](http://en.wikipedia.org/wiki/News_ticker) directly on the homescreen.

Applications are developed in the [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) language using the [Android software development kit](http://en.wikipedia.org/wiki/Android_SDK) (SDK). The SDK includes a comprehensive set of development tools, including a [debugger](http://en.wikipedia.org/wiki/Debugger), [software libraries](http://en.wikipedia.org/wiki/Software_library), a handset [emulator](http://en.wikipedia.org/wiki/Emulator) based on [QEMU](http://en.wikipedia.org/wiki/QEMU), documentation, sample code, and tutorials. The officially supported [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is [Eclipse](http://en.wikipedia.org/wiki/Eclipse_%28software%29) using the Android Development Tools (ADT) plugin.

Since Android devices are usually battery-powered, Android is designed to manage memory ([RAM](http://en.wikipedia.org/wiki/Random-access_memory)) to keep power consumption at a minimum, in contrast to desktop operating systems which generally assume they are connected to unlimited [mains electricity](http://en.wikipedia.org/wiki/Mains_electricity). When an Android app is no longer in use, the system will automatically suspend it in memory - while the app is still technically "open," suspended apps consume no resources (e.g. battery power or processing power) and sit idly in the background until needed again. This has the dual benefit of increasing the general responsiveness of Android devices, since apps don't need to be closed and reopened from scratch each time, but also ensuring background apps don't waste power needlessly.

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. For example, an application can call upon any of the phone’s core functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users. Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine that was designed to optimize memory and hardware resources in a mobile environment. Android is open source; it can be liberally extended to incorporate new cutting edge technologies as they emerge. Android provides access to a wide range of useful libraries and tools that can be used to build rich applications. For example, Android enables developers to obtain the location of the device, and allows devices to communicate with one another enabling rich peer–to–peer social applications. In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications.

Android has an active community of developers and enthusiasts who use the Android source code to develop and distribute their own modified versions of the operating system. These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, albeit without as extensive testing or quality assurance; provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the [HP Touchpad](http://en.wikipedia.org/wiki/HP_Touchpad). Community releases often come pre-[rooted](http://en.wikipedia.org/wiki/Rooting_%28Android_OS%29) and contain modifications unsuitable for non-technical users, such as the ability to [overclock](http://en.wikipedia.org/wiki/Overclock) or [over/undervolt](http://en.wikipedia.org/wiki/Dynamic_voltage_scaling) the device's processor.

Android applications run in a [sandbox](http://en.wikipedia.org/wiki/Sandbox_%28computer_security%29), an isolated area of the system that does not have access to the rest of the system's resources, unless access permissions are explicitly granted by the user when the application is installed. Before installing an application, the [Play Store](http://en.wikipedia.org/wiki/Google_Play) displays all required permissions: a game may need to enable vibration or save data to an [SD card](http://en.wikipedia.org/wiki/SD_card), for example, but should not need to read [SMS](http://en.wikipedia.org/wiki/SMS) messages or access the phonebook. After reviewing these permissions, the user can choose to accept or refuse them, installing the application only if they accept. The sandboxing and permissions system lessens the impact of vulnerabilities and bugs in applications. The open source nature of Android allows security contractors to take existing devices and adapt them for highly secure uses.

**1.1.1. Specification**

This operating system is based on version 2.6 of Linux, so it has a monolithic system kernel, what means that all system functions and drivers are grouped into one block of code.[3]

Architecture :

Android consists of five layers:

* The Linux kernel 2.6-which includes useful drivers that allow for example WiFi or Bluetooth.
* The library written in C and C + + that provide higher level functionality such as an HTML engine, or a database (SQLite).
* A runtime environment for applications based on a virtual machine, made for inefficient machines such as telephones. The aim is to translate JAVA in machine language understood by Android.
* A JAVA framework that allows applications running on the virtual machine to organize and cooperate.
* The user applications written in Java (Web browser, contact manager etc.)

**1.1.2. Characteristics**

Applications:

* Android includes most of the time many Google applications like Gmail, YouTube or Maps. These applications are delivered with the machine most of the time, except in certain cases, such as some phones running android on which the provider has replaced Google applications by its own applications.
* With android, it is possible to use widgets which are small tools that can most often get information. These widgets are directly visible on the main window.
* Android Market is an online software store to buy applications. Developers who created applications can add them into the store, and these applications can be downloaded by users, they can be both free and paid.

Multitasking:

Android allows multitasking in the sense that multiple applications can run simultaneously. With Task Manager it is possible view all running tasks and to switch from one to another easily.

SDK:

A development kit has been put at disposal of everybody. Accordingly, any developer can create their own applications, or change the android platform. This kit contains a set of libraries, powerful tools for debugging and development, a phone emulator, thorough documentation, FAQs and tutorials.

**1.2. ECLIPSE**

Eclipse is a multi-language [software development environment](http://en.wikipedia.org/wiki/Software_development_environment) comprising a base [workspace](http://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_%28computing%29) system for customizing the environment. It is written mostly in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29). It can be used to develop applications in Java and, by means of various plug-ins, other [programming languages](http://en.wikipedia.org/wiki/Programming_language). The Eclipse [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the [Eclipse Public License](http://en.wikipedia.org/wiki/Eclipse_Public_License), Eclipse [SDK](http://en.wikipedia.org/wiki/Software_development_kit) is [free and open source software](http://en.wikipedia.org/wiki/Free_and_open_source_software).

The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system. This plug-in mechanism is a lightweight [software componentry](http://en.wikipedia.org/wiki/Software_componentry) framework. In addition to allowing the Eclipse Platform to be extended using other [programming languages](http://en.wikipedia.org/wiki/Programming_language) such as [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29), the plug-in framework allows the Eclipse Platform to work with typesetting languages like [LaTeX](http://en.wikipedia.org/wiki/LaTeX), networking applications such as [telnet](http://en.wikipedia.org/wiki/Telnet) and [database management systems](http://en.wikipedia.org/wiki/Database_management_system). The plug-in architecture supports writing any desired extension to the environment, such as for [configuration management](http://en.wikipedia.org/wiki/Configuration_management). Java and [CVS](http://en.wikipedia.org/wiki/Concurrent_Versions_System) support is provided in the Eclipse [SDK](http://en.wikipedia.org/wiki/Software_development_kit), with support for other [version control systems](http://en.wikipedia.org/wiki/Version_control_system) provided by third-party plug-ins. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in [incremental](http://en.wikipedia.org/wiki/Incremental_compiler) Java compiler and a full model of the Java source files. This allows for advanced [refactoring](http://en.wikipedia.org/wiki/Refactor) techniques and code analysis. The IDE also makes use of a workspace, in this case a set of [metadata](http://en.wikipedia.org/wiki/Metadata) over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements [widgets](http://en.wikipedia.org/wiki/GUI_widget) through a widget toolkit for Java called [SWT](http://en.wikipedia.org/wiki/Standard_Widget_Toolkit), unlike most Java applications, which use the Java standard [Abstract Window Toolkit](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) or [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29). Eclipse's user interface also uses an intermediate [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) layer called [JFace](http://en.wikipedia.org/wiki/JFace), which simplifies the construction of applications based on SWT.

**1.2.1. ADT Plugin**

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give a powerful, integrated environment in which it is easy to build Android applications. ADT extends the capabilities of Eclipse to let quick set up of new Android projects, create an application UI, add packages based on the Android Framework API, debug applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications.

**1.3. ANDROID VIRTUAL DEVICE**

The Android SDK includes a mobile device emulator — a virtual mobile device that runs on your computer. The emulator lets you develop and test Android applications without using a physical device. The easiest way to create an AVD is to use the graphical [AVD Manager](http://developer.android.com/tools/devices/managing-avds.html), which you launch from Eclipse by clicking Window > AVD Manager. AVD Manager can be started from the command line by calling the android tool with the avd options, from the <sdk>/tools/ directory. AVDs can also be created on the command line by passing the android tool options. The API Level of the target is important, because your application will not be able to run on a system image whose API Level is less than that required by your application

An AVD consists of:

* A hardware profile: Defines the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.
* A mapping to a system image: You can define what version of the Android platform will run on the virtual device. You can choose a version of the standard Android platform or the system image packaged with an SDK add-on.
* Other options: You can specify the emulator skin you want to use with the AVD, which lets you control the screen dimensions, appearance, and so on. You can also specify the emulated SD card to use with the AVD.
* A dedicated storage area on your development machine: the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area.

**Chapter 2**

**FEASIBILITY**

Feasibility studies aim to objectively and rationally uncover the - strengths and weaknesses of a project, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. The analysis ensures that the proposed system is not a burden to the company. The purpose of feasibility study is not to solve the problem, but to determine whether the problem is worth solving. For feasibility analysis, some understanding of the major requirements for the system is essential.

**2.1. FEASIBILITY STUDY PROCESS**

Feasibility study comprises the following steps:-

1. Information assessment:-Identifies information about whether the system helps in achieving the objectives of the organization. It also verifies that the system can be implemented using new technology and within the budget, and whether the system can be integrated with the existing system.

2. Information collection:- Specifies the sources from where information about software can be obtained. Generally, these sources include users, and the software development team.

3. Report writing:- Uses a feasibility report, which is the conclusion of the feasibility by the software development team. In includes the recommendation whether the software development should continue or not. [4]

**2.2. IMPORTANCE OF FEASIBILITY ANALYSIS**

The purpose of a Feasibility Study is to identify the likelihood of one or more solutions meeting the stated business requirements. In other words, if you are unsure whether your solution will deliver the outcome you want, then a Project Feasibility Study will help gain that clarity. During the Feasibility Study, a variety of 'assessment' methods are undertaken. The outcome of the Feasibility Study is a confirmed solution for implementation.

**2.3. FEASIBILITY TYPES**

Following are the considerations involved in feasibility analysis:

\* Economic Feasibility

\* Technical Feasibility

\* Social Feasibility

\* Legal Feasibility

\* Operational Feasibility

**2.3.1. Economic Feasibility:**

Economic feasibility is the most frequent analysis and is carried out to evaluate the effectiveness of a new system. The amount of funds that a company can pour into the research and development of the system is limited and the expenditures must be justified. Thus the developed system must be within the budget. The technique is to compare cost of the system with its benefits; if the benefits outweigh costs, then the system is designed and implemented.[5]

Economic feasibility of the project:

a. It is a small project.

b. Hardware and software requirements are cheap.

**2.3.2 Technical Feasibility:**

Technical feasibility is analyzed to check the technical requirements of the system. Any system developed must not have a high demand of the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirement, as only minimal or null changes are required for implementing the system.

Technical feasibility of the project:

a. It can be run on any android based device.

b. It is programmed using JAVA.

**2.3.3. Social Feasibility:**

The aspect of the study is to check the impact of developed system on the society; whether the project will prove to be a part for betterment of the society or is it not safe from the perspective of security. If the project is expected to be a secure and safer one, it’s execution is proceeded.[6]

Social feasibility of the project:

a. Does not affect safety of the society.

**2.3.4. Legal Feasibility:**

The main aim of this feasibility analysis is to determine whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local Data Protection Acts.

Legal feasibility of the project:

a. No restrictions on the use of application.[7]

**2.3.5. Operational Feasibility:**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. It also considers the ease with which the user can operate the system. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and the ease of operability of the system.

Operational feasibility of the project:

a. User friendly

b. Excellent user interface [8]

**Chapter 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1. INTRODUCTION**

This document aims at defining the overall software requirements of the project ‘Calendroid’. Efforts have been made to define the requirements exhaustively and accurately. The final product will be having only features/functionalities mentioned in this document and assumptions for any additional functionality/feature should not be made.

**3.1.1. Purpose**

This specification document describes the capabilities that will be provided by the software application ‘Calendroid’. It also states the various required constraints by which the system will abide. The document is intended as a support for development, testing and users.

**3.1.2. Scope**

The software product ‘Calendroid’ is an android base application. In this application the users will be able to access calendar and set reminders, save videos and pictures specific to each date.

**3.1.3. Definitions, Acronyms and Abbreviations**

Following abbreviations have been used throughout this document:

App: Application

Sync: Synchronize

**3.1.4. Overview**

The rest part of this SRS describes the various system requirements, interfaces, features and functionality in detail.

**3.2. OVERALL DESCRIPTION**

Android is an open source [Linux](http://en.wikipedia.org/wiki/Linux)-based [operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) designed primarily for [touchscreen](http://en.wikipedia.org/wiki/Touchscreen) mobile devices such as [smartphones](http://en.wikipedia.org/wiki/Smartphone) and [tablet computers](http://en.wikipedia.org/wiki/Tablet_computer). This android app is a Calendar based application. It includes features like calendar entries, note-making, to-do list (mention the tasks which have been completed and which are yet to be completed/ also prioritize the tasks), voice remainders, diary/journal of daily activities are recorded onto the application timeline.

**3.2.1. Product Perspective**

The app will be android-based, independent software which can be used by any user having an android device.

1. System Interfaces: None
2. User Interfaces: The app will have a user-friendly interface.
3. Following screens will be provided:

* The first screen that appears for the user once the app button is clicked includes a Calendar of the current month. The user can switch between the months and years to reach any desired date.
* On clicking any of the dates, a screen opens up which contains three options for the user to choose – Journal, Camera and Video.
* The user can choose any one of the three options by simply clicking the desired one, and the respective screen would open.
* Once the user selects journal, a text area appears where reminders, important points or any other data can be saved.
* If the user chooses ‘Camera’, it would allow the user to click pictures and get them saved.
* If the last option ‘Video’ is chosen, the user can make videos and save them in phone memory.

1. Hardware Interfaces:

Any device with an android operating system will support the app.

(v) Software Interfaces:

The app requires android based phone to run. The app can also be supported on systems by using an emulator

1. Communication Interfaces:

No communication interface is required.

**3.2.2. Product Functions**

The app can be accessed by clicking on the app icon on user’s screen named as ‘Calendroid’. The app will open a screen consisting of current month’s calendar. The user can select any desired date to save data, photographs or videos on that date.

A summary of major functions that the app will perform:

1. Appears on the phone screen as an app icon named ‘Calendroid’.
2. User can access either calendar, camera, video or journal.
3. Calendar is the first scree that opens on clicking the icon. It opens date list of the current month where the user can select any date and enter reminders, to-do list or any other data for the specific date.
4. Option ‘Camera’ opens the camera of the cell phone and allows the user to click and save pictures.
5. Like camera is the ‘Video’ option which allows the user to capture and save videos.

**3.2.3. User Characteristics**

User Education and Training: -

Any person capable of using a cell phone (android based) can use the app. No special training or education is required to make user familiar with the concept.

**3.2.4. Constraints**

The app is functional only in android environment with API level of 10 or above.

**3.2.5. Assumptions and Dependencies**

* The device is an android based phone.
* User is aware of android environment and is capable of using android based apps.
* The operating system has an API level of 10 or above.
  + 1. **Apportioning of Requirements**

Not Required

**3.3.SPECIFIC REQUIREMENTS**

**3.3.1. Functional requirements**

The various Functional requirements of the system can be summarized as follows:

* Android environment is essential to run the app.
* Development of the app requires Eclipse with a plug in for SDK.
* Functional camera within the phone is required.

**3.3.2. Performance Requirements**

Considering the interactive nature of the task the system must have the following characters.

* Minimum response time
* Less Memory space
* High reliability
* High flexibility
* User friendly

**3.3.3. Non Functional Requirements**

**Performance Requirements**

A cellular phone (with camera) running on Android is required for the application to run.

**Other Non Functional Requirements**

Non-functional requirements define system properties and constraints it arises through user needs, because of budget constraints or organizational policies, or due to the external factors such as safety regulations, privacy registration and so on. Non-functional requirements are:

· Security

· Reliability

· Maintainability

· Portability

· Extensibility

· Application Affinity/Compatibility

· Resource Utilization

* 1. **SOFTWARE SYSTEM ATTRIBUTES**
     1. **Security**

The app has no security issues as it does not ask for any person information from the user. Mobile users can simply download the app in their android phones and get started with it.

* + 1. **Maintainability**

The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the product.

* + 1. **Portability**

Application will be easily portable on any android-based device.

**Chapter 4**

**DESIGN PHASE**

**4.1. USE CASE DIAGRAM**

**4.1.1. What is Use Case**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal.

Use cases can be employed during several stages of software development, such as planning system requirements, validating design, testing software, and creating an outline for online help and user manuals.

**4.1.2. What does a use case describe**

A use case describes a sequence of interactions between a user and a Web site, without specifying the user interface.

Each use case captures:

* The actor (who is using the Web site?)
* The interaction (what does the user want to do?)
* The goal (what is the user's goal?)[9]

**4.1.3. Characteristics**

A use case (or set of use cases) has these characteristics:

* Organizes functional requirements
* Models the goals of system/actor (user) interactions
* Records paths (called scenarios) from trigger events to goals
* Describes one main flow of events (also called a basic course of action), and possibly other ones, called exceptional flows of events (also called alternate courses of action)
* Is multi-level, so that one use case can use the functionality of another one.[10]

**4.1.4. How to Write Use Case:**

Generally, the steps in a use case are written in an easy-to-understand narrative. This engages members of the design team and encourages them to be actively involved in defining the requirements.

Kenworthy (1997) outlines eight steps to developing use cases:

1. Identify who is going to be using the system.
2. Pick one of those actors.
3. Define what that actor wants to do on the system. Each thing the actor does on the system becomes a use case.
4. For each use case, decide on the normal course of events when that actor is using the system.
5. Describe the basic course in the description for the use case. Describe it in terms of what the actor does and what the system does in response that the actor should be aware of.
6. When the basic course is described, consider alternate courses of events and add those to "extend" the use case.
7. Look for commonalities among the use cases. Extract these and note them as common course use cases.
8. Repeat the steps 2 through 7 for all other actors.

**4.1.5. Use Case for Calendroid**

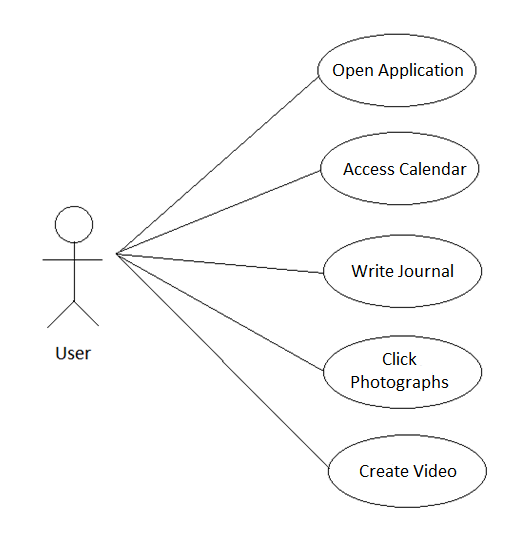
****

Fig4.1. *Use Case Diagram*

**4.2. DATA FLOW DIAGRAM**

**4.2.1. What are DFDs:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system), modeling its process aspects. DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.[11]

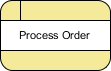
**4.2.2. How to draw DFDs:**

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people use to draw external entities on the edge of a diagram.

notation (enternal entity)

**Process**

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.



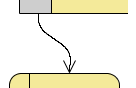
**Data Store**

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc.

notation (data store)

**Data Flow**

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.



**4.2.3. 0-Level DFD**

A context (level 0) diagram documents the system’s boundaries by highlighting its sources and destinations. Documenting the system’s boundaries by drawing a context diagram helps the analyst, the user, and the responsible managers visualize alternative high-level logical system designs. Moving the boundaries significantly changes the system, and the ability to visualize the implications of different boundary assumptions is a powerful reason for creating a context diagram.

**4.2.4. 1-Level DFD**

A level 1 data flow diagram shows the system’s primary processes, data stores, sources, and destinations linked by data flows. Generally, a system’s primary processes are independent, and thus, separated from each other by intermediate data stores that suggest the data are held in some way between processes. A level 1 process is a composite item that might incorporate related programs, routines, manual procedures, hardware-based procedures, and other activities. As the data flow diagram is decomposed, the various sub-processes are eventually isolated and defined. [12]

**4.2.5. DFD for Calendroid**

MEMORY

Photographs/ video/ Save data

INPUT

Text

Fig4.3. *Level 0 DFD for Calendroid*

Request Display

CALENDAR

USER

Display

List

DATE

Display

Select Select Select

VIDEO

PHOTOGRAPH

JOURNAL

Take Photo Shoot video

Write text

Data

MEMORY

Fig4.4. *Level 1 DFD for Calendroid*

**4.3. ENTITY-RELATIONSHIP DIAGRAM**

**4.3.1. What is ER Diagram**

It is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of [data](http://www.webopedia.com/TERM/D/data.html) within [databases](http://www.webopedia.com/TERM/D/database.html) or information systems. An entity is a piece of data-an [object](http://www.webopedia.com/TERM/O/object.html) or concept about which data is stored. A relationship is how the data is shared between entities. There are three types of relationships between entities:

* **one-to-one**: one instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).
* **one-to-many**: one instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.
* **many-to-many**: one instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.

**4.3.2. ER Notations**

[Peter Chen](http://www.csc.lsu.edu/%7Echen/chen.html) developed ERDs in 1976. Since then Charles Bachman and James Martin have added some sligh refinements to the basic ERD principles.

##### Entity

An entity is an object or concept about which you want to store information.  
[Learn how to edit text on an entity.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)

Entity

##### Weak Entity

A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Weak Entity

##### Key attribute

A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

Key attribute

##### Multivalued attribute

A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.

Multivalued attribute

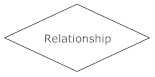
##### Derived attribute

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

Derived attribute

##### Relationships

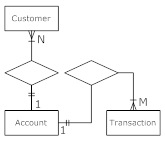
Relationships illustrate how two entities share information in the database structure.



##### Cardinality

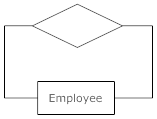
Cardinality specifies how many instances of an entity relate to one instance of another entity.

Ordinality is also closely linked to cardinality. While cardinality specifies the occurences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.



##### Recursive relationship

In some cases, entities can be self-linked. For example, employees can supervise other employees.



**4.3.3. ER Diagram for Calendroid**

user

Choose Date

Date > Present Date?

Yes

Select Option

Journal

No

Read Only

Video

Camera

Read/Write

Fig4.3. *E-R Diagram*

**Chapter 5**

**IMPLEMENTATION**

**5.1 TECHNOLOGY USED**

The technology used for implementing Calendroid was Java in Eclipse. Eclipse is used to provide an android-like environment. The software is also used to create an android virtual machine which brings android phone effects on the system. The application is tested on the same platform and has been snapshotted.

Java

Version Used: jdk1.7.0

**5.1.1. ABOUT JAVA**

Java is a programming language originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent,class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is as of 2012 one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.[13]

New features in jdk 1.7 as compared to previous versions

\* Strings in switch statements

\* Better integral literals

\* Multi-catch exceptions, which we cover here

\* Improved type inference for generic instance creation ("diamond")

\* try-with-resources, which we cover here

\* Simplified varargs method invocation [14]

For database storage, we have created a dialog box which accesses all files from the computer system we are working on. The image in which the data is to be encrypted can be browsed and then the encrypted image is saved on the system. While decrypting, the encrypted image can be retrieved from the system.

**5.2 CLASSES USED**

**5.2.1. Database.java**

It is the first class of the coding. As the name says, it is used to create database in order to save the content captured by the user. The class also handles updates in the database by appending the already existing content to the newly added data.

**5.2.2. DisplayMenu.java**

The display menu is a window that shows the chosen date and three options, each depicted by an image. The three options are : diary, camera and video. If the user clicks on ‘diary’, the class creates a file to be written in. If the user clicks on camera, the class calls photos.class and if the user chooses video, the class calls videos.class.

The DisplayMenu window appears as:



Fig5.1. *Display Menu*

**5.2.3. Photos.java**

The class represents a window which gives the choice of either traversing the gallery or to open camera. When the user clicks on the camera icon, the application opens the camera of the cell phone; allowing the user to click pictures.

The Photos window appears as :

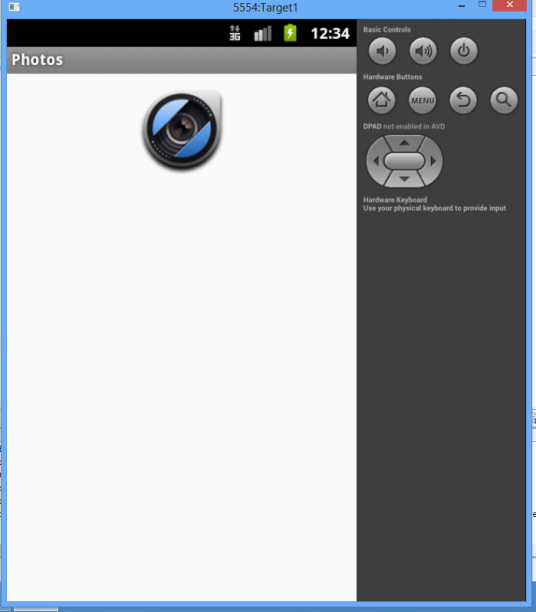


Fig5.2. *Photos Window*

**5.2.4. Reader.java**

Read.java is a class that enables user to read the journal of a particular date. If the user has already written content corresponding to a specific date then, revisiting the same date allows the user to read the content. Also, if the user visits a previous date then he is disabled of writing any content. Rather the user can only read the contents of the file.

The Reader window appears as:

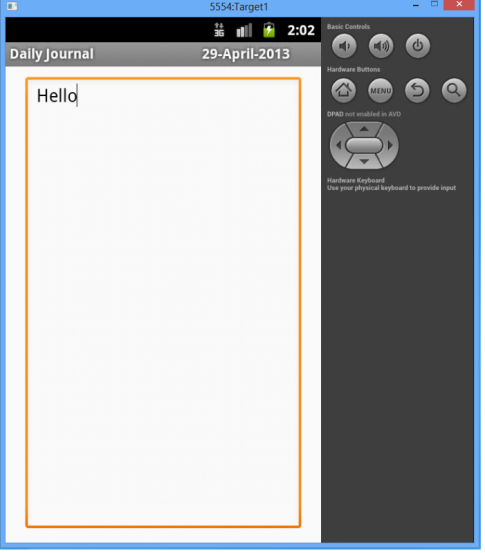


Fig5.3. *Reader window*

**5.2.5. SimpleCalendarViewActivity.java**

The class as the name tells, shows the calendar view. The first window that appears on clicking the application is the SimpleCalendarViewActivity window. It depicts the calendar of the current month. User can switch between different months by using the arrow heads provided on the screen. The class calculates the month and year on each click and comes up with the correct month’s calendar. The calendar is shown in pink colour where all the dates appear in white colour. The present date is shown in blue colour and the inaccessible dates appear in grey colour.

The SimpleCalendarViewAcitivity Window appears as:

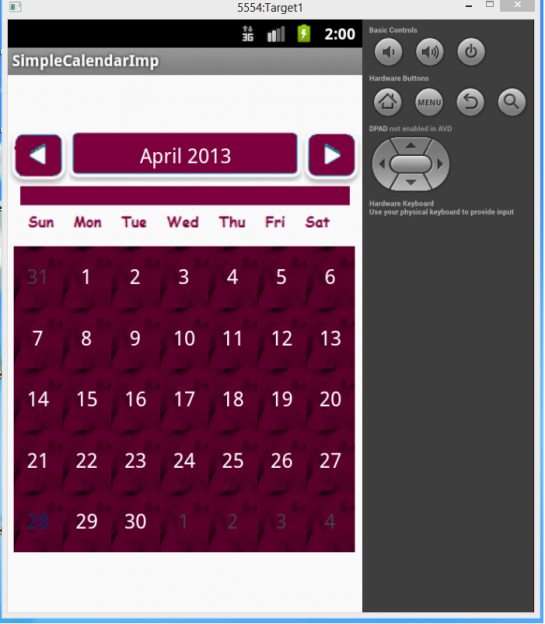


Fig5.4. *Calendar Window*

**5.2.6. Videos.java**

The videos.java class is same as photos.java. Just as the photos class gives an option between gallery and camera, similarly the video class gives the options of either going through the gallery or operating the camera in video mode.

The Videos.java window appears as:

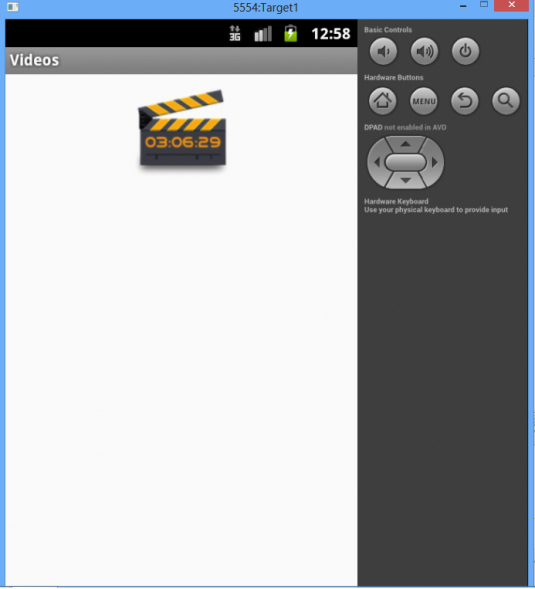


Fig5.5. *Videos Window*

**5.2.7. Writer.java**

The writer.java class opens up a writer window. The window appears once the user chooses the ‘journal’ option from the display menu. It is a blank window with a blinking cursor that enables the use to type text. The typed text stayed saved in the database without the user requiring to save it explicitly. Whenever the user visits the same date’s journal again, the text appears as it was typed.

The Writer window appears as:

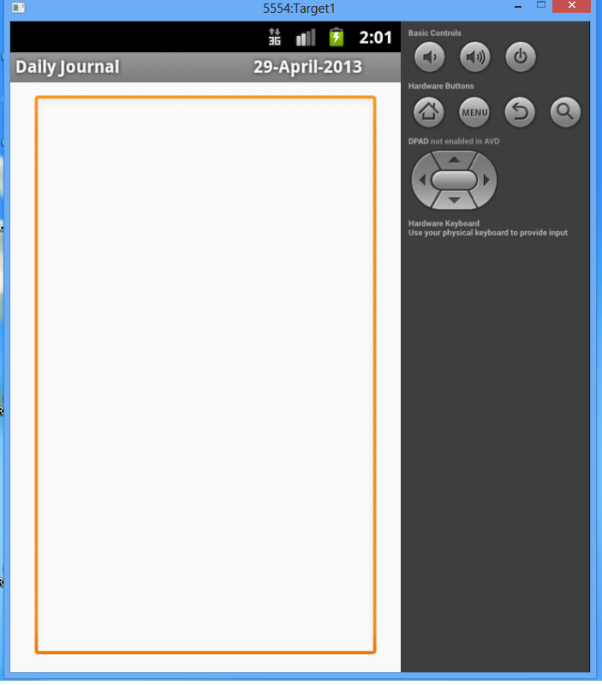


Fig5.6. *Writer Window*

**Chapter 6**

**TESTING**

##### Testing is an activity that is used to discover errors and correct them, so that we are able to create a defect-free product for our customer. Testing is an important stage which follows the Coding stage in the software development life cycle. The objective of testing is to evaluate if we have created the system correctly. During the earlier stages, the focus was to check what is being built but in testing when we have the end product ready, our focus shifts to validate whether the product that has been built has been built correctly or not. Hence, the focus shifts from building the product right to building the right product. Testing can also be defined as the process of executing a program with the specific intent of finding an error. Success of a test is determined by the number of errors it has uncovered. Tests can be conducted by the developer or by an independent testing team. What one should remember is that the role of a good tester is to show the presence of the defects or errors of that software.[16]

Creation of the test strategy is the first step. It is based on the Requirements Document, the Functional Specifications Document, and the Design Document. The test strategy describes the overall plan and approach to be taken for testing, the deliverables, and the process for test reporting. The next step is to create the test cases, containing the individual scenarios, which would be tested with their expected outcomes. Test cases are executed by the tester and results of the tests are documented in the test log. The defects of testing are recorded in defect-tracking tools such as the internal tool Prolite or the external tool called Test Director, depending on the requirement of a project. The owner of the application being tested then updates the application, closes the defects, and updates Prolite with the defect status in the tools. Re-testing may be conducted to verify closure of defects.

##### Testing defines the status of the working functionalities of any particular system. Through testing particular software one can’t identify the defects in it but can analyse the performance of software and its working behavior. By testing the software we can find the limitations that become the conditions on which the performance is measured on that particular level.[17**]**

The main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running indifferent operating environments.

**6.1. ARTEFACTS OF TESTING**

For testing the application the testing process produces several artefacts.

The different artefacts are:

Test plan: Test plan gives us the processes of testing we subject the application, called as the test process. The developers execute the test plan and the results are used for the purpose of management and future developments.

Traceability matrix: A traceability matrix is a table that links the design documents to the text documents. This also changes the test processes when the source documents are changed.

Test case: A test case consists of a unique identifier, identifies the requirements of the project from the design phase, have information about the series of steps like input, output, expected result and actual result. The series of steps are stored in text document or excel spread sheet.

Test script: A test script is a product of work generated by automated regression tools. It is a combination of test case, test procedure and test data.

Test suite: A test suite is a collection of test cases. The test suite consists of the detailed instructions and the role of each collection of test cases.

Test data: The place where the test values and components can be modified is known as Test data.

Test harness: A test harness is a collection of software, inputs, outputs and configurations used for the application

**6.2. TYPES OF TESTING**

**6.2.1. Performance testing**

Performance testing is designed to test run time performance of software within the context of an integrated system. It is not until all systems elements are fully integrated and certified as free of defects the true performance of a system can be ascertained.

**Performance factors**

Important influence factors to the performance of a Java program can be separated into two main parts:

* Memory Consumption of the Java program
* Total runtime of a program

## Memory

Java handles its memory in two areas: The heap and the stack.

### Memory Consumption

The total used / free memory of an program can be obtained in the program via java.lang.Runtime.getRuntime();

### Runtime of a Java program

We use System.currentTimeMillis() to get the start time and the end time and calculate the difference.

**6.2.2. Security testing**

If your site requires firewalls, encryption, user authentication, financial transactions, or access to databases with sensitive data, you may need to test these and also test your site’s overall protection against unauthorized internal or external access.

There are four Basic Concepts of Security Testing

* Confidentiality

A security measure which protests against the disclosure of information to parties other then the intended recipient.

* Integrity

A measure intended to allow the receiver to determine that the information which it receives has not been altered in transit or by other than the originator of the information.

* Authentication

A measure designed to establish the validity of a transmission, message or originator and allows a receiver to have confidence that information it receives originated from a specific known source.

* Authorization

The process of determining that a requester is allowed to receive a service or perform an operation.

Some of the causes of Security failure:

* Password Management Flaws The computer user uses weak passwords that could be discovered by brute force.
* Software Bugs – The programmer leaves an exploitable bug in a software program. The software bug may allow an attacker to misuse an application through (for example) bypassing access control checks.
* Unchecked User Input – The program assumes that all user input is safe. Programs that do not check user input can allow unintended direct execution of commands or SQL statements (known as SQL injection or other non-validated inputs).

**6.2.3. Exploratory Testing**

Often taken to mean a creative, internal software test that is not based on formal test plans or test cases; testers may be learning the software as they test it.

**6.2.4. Functional Testing (or) Business functional testing**

All the functions in the applications should be tested against the requirements document to ensure that the product conforms with what was specified.(They meet functional requirements). Business functions are generally defined in the requirements Document. Each business function has certain rules, which can’t be broken. Whether they applied to the user interface behavior or data behind the applications. Both levels need to be verified. Business functions may span several windows (or) several menu options, so simply testing that all windows and menus can be used is not enough to verify the business functions. You must verify the business functions as discrete units of your testing

\* Study SRS

\* Identify Unit Functions

\* For each unit function

\* Take each input function

\* Identify Equivalence class

\* Form Test cases

\* Form Test cases for boundary values

\* From Test cases for Error Guessing

\* Form Unit function v/s Test cases, Cross Reference Matrix

**6.2.5. User Interface Testing (or) structural testing**

It verifies whether all the objects of user interface design specifications are met. It examines the spelling of button test, window title test and label test. Checks for the consistency or duplication of accelerator key letters and examines the positions and alignments of window objects

**6.2.6. Volume Testing**

Testing the applications with voluminous amount of data and see whether the application produces the anticipated results (Boundary value analysis)

**6.2.7. Stress Testing**

Testing the applications response when there is a scarcity for system resources

**6.2.8. Load Testing**

It verifies the performance of the server under stress of many clients requesting data at the same time.

**6.2.9. Installation testing**

The tester should install the systems to determine whether installation process is viable or not based on the installation guide

**6.2.10. Configuration Testing**

The system should be tested to determine it works correctly with appropriate software and hardware configurations

**6.2.11. Compatibility Testing**

The system should be tested to determine whether it is compatible with other systems (applications) that it needs to interface with

**6.2.12. Documentation Testing**

It is performed to verify the accuracy and completeness of user documentation 1. This testing is done to verify whether the documented functionality matches the software functionality 2. The documentation is easy to follow, comprehensive and well edited if the application under test has context sensitive help, it must be verified as part of documentation testing

**6.2.13. Recovery/Error Testing**

Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems

**6.2.14. Comparison Testing**

Testing that compares software weaknesses and strengths to competing products

**6.2.15. Acceptance Testing**

Acceptance testing, which is black box testing, will give the client the opportunity to verify the system functionality and usability prior to the system being moved to production. The acceptance test will be the responsibility of the client; however, it will be conducted with full support from the project team. The Test Team will work with the client to develop the acceptance criteria.

**6.2.16. Alpha Testing**

Testing is an application when development is nearing completion, Minor design changes may still be made as a result of such testing. Alpha Testing is typically performed by end-users or others, not by programmers or testers

**6.2.17. Beta Testing**

Testing when development and testing are essentially completed and final bugs, problems need to be found before the final release. Beta Testing is typically done by end-users or others, not by programmers or testers.

**6.2.18. Validation Testing**

Validation is the process of finding whether the product is built correct or not. The software application or product that is designed should fulfill the requirements and reach the expectations set by the user. Validation is done while developing or at the final stage of development process to determine whether it is satisfies the specified requirements of user.

**6.2.19. Output Testing**

After completion of validation testing the next process is output testing. Output testing is the process of testing the output generated by the application for the specified inputs. This process checks weather the application is producing the required output as per the user’s specification or not.

**6.2.20. Regression Testing**

The objective of regression testing is to ensure software remains intact. A baseline set of data and scripts will be maintained and executed to verify changes introduced during the release have not “undone” any previous code. Expected results from the baseline are compared to results of the software being regression tested. All discrepancies will be highlighted and accounted for, before testing proceeds to the next level

**6.2.21. Incremental Integration Testing**

Continuous testing of an application as new functionality is recommended. This may require various aspects of an application’s functionality be independent enough to work separately before all parts of the program are completed, or that test drivers are developed as needed. This type of testing may be performed by programmers or by testers

**6.2.22. Usability Testing**

Testing for ‘user-friendliness’ clearly this is subjective and will depend on the targeted end-user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers

**6.2.23. Unit Testing**

Unit testing is the approach of taking a small part of testable application and executing it according to the requirements and testing the application behavior. Unit testing is used for detecting the defects that occur during execution.

When an algorithm is executed, the integrity should be maintained by the data structures. Unit testing is made use for testing the functionality of each algorithm during execution.

**6.2.24. Integration Testing**

Upon completion of unit testing, integration testing, which is black box testing, will begin. The purpose is to ensure distinct components of the application still work in accordance to customer requirements. Test sets will be developed with the express purpose of exercising the interfaces between the components. This activity is to be carried out by the Test Team. Integration test will be termed complete when actual results and expected results are either in line or differences are explainable/acceptable based on client input

**6.2.25. System Testing**

Upon completion of integration testing, the Test Team will begin system testing. During system testing, which is a black box test, the complete system is configured in a controlled environment to validate its accuracy and completeness in performing the functions as designed. The system test will simulate production in that it will occur in the “production-like” test environment and test all of the functions of the system that will be required in production. The Test Team will complete the system test. Prior to the system test, the unit and integration test results will be reviewed by SQA to ensure all problems have been resolved. It is important for higher level testing efforts to understand unresolved problems from the lower testing levels. System testing is deemed complete when actual results and expected results are either in line or differences are explainable/acceptable based on client input

**6.2.26. Parallel/Audit Testing**

##### Testing where the user reconciles the output of the new system to the output of the current system to verify the new records.

**Chapter 7**

**SCOPE OF THE WORK**

This project focuses on creating an android based application capable of saving data corresponding to a specific date. Usually, the user is able to save only text corresponding to a date; also known as reminders. The project is aimed at allowing the user to store photographs as well as videos as well on a specific date. On clicking the application icon on the phone’s screen, a calendar window appears. The window shows the calendar dates of current month. User is free to switch to different months or years in order to access any date of his choice. Once the user clicks on the desired date, a window showing three icons appear: Diary, Camera and Video. User can choose any of the three options. If the user clicks on diary, a writing window appears. The user can type any reminders, work list or any data of his choice. The written text gets saved simultaneously in an automatically created file, named after the calendar date. If the user chooses camera option, the camera of the phone opens. User can click photographs which get saved on the calendar date chosen. Similarly, if the user opts for video option, the camera opens in video mode. User can shoot videos which get saved corresponding to the selected date. All these features are enabled only if the user selects a date which is greater than the present day’s date. In case the user selects any of the previous dates, all the three options are read only. This means that the user can choose diary option, but can only read the pre written data. The user cannot add data or make changes.

**7.1 SCOPE**

The application allows the user to store data easily. Any of the information types, be it text, photographs or videos, can be saved in correspondence to a specific date in calendar. The application also allows automatic saving of the content. The user need not click any ‘save button’ after completing the job. Anything added to the application gets automatically saved in a file that is named after the calendar’s date. The scope of project lies within the platform for which it is created – Android. With the increase in the number and make of mobile phones, there comes a demand for better applications. And in turn, huge scope of android mobile application development in India. Open source code and licensing of Android allows the developers and device manufacturers to modify the software according to their needs. Android platform has brought about cutting-edge technologies in app development. Before the acceptance of Android, the mobile app development industry was dominated by Proprietary OS like Symbian and iOS. With Android, came the option for dynamic app development at a lower cost. When thinking of the scope of Android Application Development in India, we can take these three primary notions into consideration:

• Revenue – The need for inventive App Developers are increasing in the current job market. Mobile application development can also be taken up as a part time job, where one can create applications and submit it to the Google Play store which can be downloaded. Google adsense ads can be displayed in your application which again provides monetary gains.

• Ease of use – Learning Android Programming is fairly easy and app development is cost effective. Any software developer who can think out of the box will be able to put Android into extraordinary use.

• Support – The most important attraction of Android is backing by Google.

**Chapter 8**

**DISCUSSION AND RESULT**

**8.1 INTRODUCTION**

One of the strengths of the Android platform compared to others is that it has an open source basis, which makes it easier to produce applications and distribute them without waiting for a lengthy approval process. There are two techniques that can be used to produce Android applications with a PC. The first uses the Android Software Development Kit (SDK). This allows writing raw code and helps in getting it working in the Android environment. The second uses App Inventor, a Google Labs tool that's still in beta. Requirements to start an Android application building:

* Java Development Kit
* Eclipse Integrated Development Environment
* Android SDK

An Android application is a collection of tasks, each of which is called an activity. Each activity within an application has a unique purpose and user interface. This project implements a framework which is intended to provide an android application capable of saving reminders, to-do list, calendar entries, note-making, photographs and videos on a specific date.

**8.2 JAVA DEVELOPMENT KIT**

The Java Development Kit (JDK) is an implementation of either one of the [Java SE](http://en.wikipedia.org/wiki/Java_SE), [Java EE](http://en.wikipedia.org/wiki/Java_EE) or [Java ME](http://en.wikipedia.org/wiki/Java_ME) platforms released by [Oracle Corporation](http://en.wikipedia.org/wiki/Oracle_Corporation) in the form of a binary product aimed at [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) developers on [Solaris](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29), [Linux](http://en.wikipedia.org/wiki/Linux), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) or [Windows](http://en.wikipedia.org/wiki/Windows).[[2]](http://en.wikipedia.org/wiki/Java_Development_Kit#cite_note-2) Since the introduction of [Java](http://en.wikipedia.org/wiki/Java_%28software_platform%29) platform, it has been by far the most widely used Software Development Kit. The JDK comes with a complete [Java Runtime Environment](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Internationalization_and_localization) libraries and the [IDL](http://en.wikipedia.org/wiki/Interface_description_language) libraries. The JDK has as its primary components a collection of programming tools, including:

* [appletviewer](http://en.wikipedia.org/wiki/AppletViewer) – this tool can be used to run and debug Java applets without a web browser
* apt – the [annotation-processing tool](http://en.wikipedia.org/wiki/Metadata_facility_for_Java)
* extcheck – a utility which can detect JAR-file conflicts
* idlj – the IDL-to-Java compiler. This utility generates Java [bindings](http://en.wikipedia.org/wiki/Language_binding) from a given [Java IDL](http://en.wikipedia.org/wiki/Java_Interface_Definition_Language) file.
* java – the [loader](http://en.wikipedia.org/wiki/Loader_%28computing%29) for Java applications. This tool is an interpreter and can interpret the class files generated by the [javac](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Javac) – the [Java compiler](http://en.wikipedia.org/wiki/Java_compiler), which converts source code into [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode)
* [javadoc](http://en.wikipedia.org/wiki/Javadoc) – the documentation generator, which automatically generates documentation from [source code](http://en.wikipedia.org/wiki/Source_code) comments

**8.3 ECLIPSE IDE**

Eclipse is a multi-language [software development environment](http://en.wikipedia.org/wiki/Software_development_environment) comprising a base [workspace](http://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_%28computing%29) system for customizing the environment. It is written mostly in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29). Development environments include the Eclipse Java development tools (JDT). The initial [codebase](http://en.wikipedia.org/wiki/Codebase) originated from [IBM VisualAge](http://en.wikipedia.org/wiki/IBM_VisualAge). The Eclipse [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages. The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system, in contrast to some other applications, in which functionality is [hard coded](http://en.wikipedia.org/wiki/Hard_code). This plug-in mechanism is a lightweight [software componentry](http://en.wikipedia.org/wiki/Software_componentry) framework. In addition to allowing the Eclipse Platform to be extended using other [programming languages](http://en.wikipedia.org/wiki/Programming_language) such as [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29), the plug-in framework allows the Eclipse Platform to work with typesetting languages. The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in [incremental](http://en.wikipedia.org/wiki/Incremental_compiler) Java compiler and a full model of the Java source files.

**8.4 ANDROID SDK**

Android software development is the process by which new applications are created for the [Android operating system](http://en.wikipedia.org/wiki/Android_%28operating_system%29). Applications are usually developed in the [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) programming language using the Android Software Development Kit. The Android [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK) includes a comprehensive set of development tools.[[7]](http://en.wikipedia.org/wiki/Android_software_development#cite_note-7) These include a [debugger](http://en.wikipedia.org/wiki/Debugger), [libraries](http://en.wikipedia.org/wiki/Software_library), a handset [emulator](http://en.wikipedia.org/wiki/Emulator) based on [QEMU](http://en.wikipedia.org/wiki/QEMU), documentation, sample code, and tutorials. Currently supported development platforms include computers running [Linux](http://en.wikipedia.org/wiki/Linux_kernel) (any modern desktop [Linux distribution](http://en.wikipedia.org/wiki/List_of_GNU/Linux_distributions)), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) 10.5.8 or later, [Windows XP](http://en.wikipedia.org/wiki/Windows_XP) or later. The officially supported [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is [Eclipse](http://en.wikipedia.org/wiki/Eclipse_%28software%29) using the Android Development Tools (ADT) Plugin. 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.

**8.5 RESULT**

Our framework efficiently works in providing a calendar environment capable of storing data in the form of text, photographs or videos. The calendar allows the user to store data on a specific date so that the user may later check the date to retrieve the data. In today’s world where Android is a growing market and is giving tough competition to all other systems due to it’s open source nature, a useful android application can prove to be a help for everyone. In the busy schedule, people usually find it difficult to keep a track of activities. Hence, the application will be a great help to save the entire to-do list, reminders for future reference. This enables the user to save the tasks corresponding to a specific date. Hence, the user can save the task on the date on which it is to be done. Also, everything added to the list gets automatically saved in the memory in the name of the calendar date, corresponding to which the text is typed.

**REFERENCES**

[1]http://www.programmer2programmer.net/live\_projects/project\_7/calendar.asx

[2] http://1000projects.org/final-year-project-on-android

[3] http://www.101seminartopics.com/?s=calendar&submit

[4] <http://en.wikipedia.org/wiki/Feasibility_study>

[5] <http://www.authorstream.com/Presentation/aSGuest91848-916487-introduction/>

[6] <http://infochk.wordpress.com/2011/02/23/calendar/>

[7] <http://www.method123.com/feasibility-study.php>

[8]<http://www.indiastudychannel.com/resources/102399FeasibilityTypesFesibility.aspx>

[9] <http://searchsoftwarequality.techtarget.com/definition/use-case>

[10] http://www.usability.gov/methods/usecases.html

[11] <http://en.wikipedia.org/wiki/Data_flow_diagram>

[12] <http://www.hit.ac.il/staff/leonidm/information-systems/ch24.html>

[13] <http://en.wikipedia.org/wiki/Java_(programming_language>

[14] <http://www.oracle.com/technetwork/articles/java/java7exceptions-486908.html>

[15] http://en.wikipedia.org/wiki/Eclipse

[16]<http://rajeevprabhakaran.wordpress.com/2008/11/20/different-types-of-testing/>

[17] http://techforum4u.com/content.php/417-Testing-In-Software-Engineering