GOVERNMENT POLYTECHNIC COLLEGE KUNNAMKULAM

KIZHOOR (P.O), THRISSUR (DT), KERALA,

PIN-680523

PROJECT REPORT

2019-2020

PROTECTION SYSTEM

Presented by

AKSHAY M.V
AMAL DEV V.D
JITHEESH. E
RAMSHAD N.E
APPU S

Directorate of Technical Education

Government of Kerala

GOVERNMENT POLYTECHNIC COLLEGE KUNNAMKULAM

KIZHOOR (P.O), THRISSUR (DT), KERALA,

PIN-680523

CERTIFICATE

This is to certify that the project report titled **"PROTECTION SYSTEM"** was presented by this group in partial fulfillment of the requirement for the award of Diploma in Computer Engineering under the Technical Education Department during the academic year 2019-2020 at **Govt. Polytechnic College, Kunnamkulam**

Date:

Place:

Guide Head of the Department

Internal Examiner External Examiner

Acknowledgement

At first, I like to express my indebtedness to the Almighty God who showered his abundant grace on me to make this project a success. I wish to express my sincere gratitude to **Seema K.N**, our Principal to provide us all necessary facilities. I take this opportunity to express my thanks to **Seena I T**, Head of the department of **Computer Engineering** for her valuable guidance and support to shape this program in a systematic way. I am extremely thankful to our respected lecturers for their keen interest and valuable suggestions in this programme. In addition to this I would like to thank all the staff members of Computer Department for their suggestions and constructive critics. And finally I use this opportunity to express my sincere gratitude towards my parents and all my friends for their support.

ABSTRACT

It is an android application for mobile devices .In todays world, people using smart phone have increased rapidly and hence,a smart phone can be used effciently for personal security or various other protection purposes.The heinous incident that outraged the entire nation have waken us to go for the safety issues and so a host of new apps have been developed to provide security system to women via their phones.This Android Application for the safety of women and this app can be activated this app by a single click, whenever need arises.

ADVANTAGES

- This application used to protect women.
- It is a user friendly application
- This application reciver can locate the currectlocation of User..

Disadvantage

- If there is no Internet connection,the message and location will not be send.

Table of Contents

CHAPTER 1 :

INTRODUCTION

1.1 PROJECT PLANS

1.1.1 ABOUT THE PROJECT

Android application for the reciver to acces current location a user.

This android application for the safety of women and this app can be Activated this app by a single click,When ever need arises.

A single click on this app identifies the location of place through GPS And sends a message comparing this location URL to the registered Contacts and also call on the first registered contact to help the one in Dangerous situvations.

This application have php database to store the details of user.

1.1.2 PURPOSE AND SCOPE

The application keeps the details Of user secured . This application is eassy To use the user.The unique feature of this application is to send the message to the registered contact continuously for every five minutes until the “STOP” button in The application is clicked.continuous location tracking information via sms helps To find the location of the victim quickly and can be resued safely.

1.2 ABOUT ANDROID

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touch screen mobile devices such as Smartphone's and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 8.1 "Oreo", released in December 2017. The core Android source code is known as Android Open Source Project (AOSP), and is primarily licensed under the Apache License. Android is also associated with a suite of proprietary software developed by Google, including core apps for services such as Gmail and Google Search, as well as the application store and digital distribution platform Google Play, and associated development platform. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems, such as Amazon.com's Fire OS, which utilize its own equivalents to these Google Mobile Services. Android has been the best-selling OS worldwide on smart phone's since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active

users, the largest installed base of any operating system, and as of 2017, the Google Play store features over 3.5 million apps. Google introduced the Pixel and Pixel XL smart phones in October 2016, marketed as being the first phones made by Google, and exclusively featured certain software features, such as the Google Assistant, before wider rollout. The Pixel phones replaced the Nexus series, with a new generation of Pixel phones launched in October 2017. FEATURES OF OS

1.INTERFACE: Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard.Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented.Android devices boot to the homescreen, the primary navigation and information "hub" on Android devices, analogous to the desktop found on personal computers. 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 a weather forecast.Along the top of the screen is a status bar, showing information about the device and its connectivity.

This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates.

Notifications are "short, timely, and relevant information about your app when it’s not in use", and when tapped, users are directed to a screen inside the app relating to the notification.

An All Apps screen lists all installed applications, with the ability for users to drag an app from the list onto the home screen. A Recent screen lets users switch between recently used apps

2.APPLICATIONS: Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support. The Go programming language is also supported, although with a limited set of application programming interfaces (API). In May 2017, Google announced support for Android app development in the Kotlin programming language.

The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android

application development. Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices.

3. MEMORY MANAGEMENT Since Android devices are usually battery-powered, Android is designed to manage processes to keep power consumption at a minimum. When an application is not in use the system suspends its operation so that, while available for immediate use rather than closed, it does not use battery power or CPU resources. Android manages the applications stored in memory automatically: when memory is low, the system will begin invisibly and automatically closing inactive processes, starting with those that have been inactive for the longest amount of time.

1.2.1 ANDROID DEVELOPMENT

Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. Rubin described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences". The early intentions of the company were to develop an advanced operating system for digital cameras, and this was the basis of its pitch to investors in April 2004. The company then decided that the market for cameras was not large enough for its goals, and by five months later it had diverted its efforts and was pitching Android as a handset operating system that would rival Symbian and Microsoft Windows Mobile. Rubin had difficulty attracting investors early on, and Android was facing eviction from its office space. Steve Perlman, a close friend of Rubin, brought him $10,000 in cash in an envelope, and shortly thereafter wired an undisclosed amount as seed funding. Perlman refused a stake in the company, and has stated "I did it because I believed in the thing, and I wanted to help Andy." In July 2005, Google acquired Android Inc. for at least $50 million. Its key employees, including Rubin, Miner and White, joined Google as part of the acquisition. Not much was known about the secretive Android at the time, with the company having provided few details other than that it was making software for mobile phones. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradeable system. Google had "lined up a

series of hardware components and software partners and signaled to carriers that it was open to various degrees of cooperation". Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. An early prototype had a close resemblance to a BlackBerry phone, with no touch screen and a physical QWERTY keyboard, but the arrival of 2007's Apple iPhone meant that Android "had to go back to the drawing board". Google later changed its Android specification documents to state that "Touchscreens will be supported", although "the Product was designed with the presence of discrete physical buttons as an assumption, therefore a touchscreen cannot completely replace physical buttons". By 2008, both Nokia and BlackBerry announced touch-based smartphones to rival the iPhone 3G, and Android's focus eventually switched to just touchscreens. The first commercially available smartphone running Android was the HTC Dream, also known as TMobile G1, announced on September 23, 2008. On November 5, 2007, the Open Handset Alliance, a consortium of technology companies including Google, device manufacturers such as HTC, Motorola and Samsung, wireless carriers such as Sprint and T-Mobile, and chipset makers such as Qualcomm and Texas Instruments, unveiled itself, with a goal to develop "the first truly open and comprehensive platform for mobile devices".Within a year, the Open Handset Alliance faced two other open source competitors, the Symbian Foundation and the LiMo Foundation, the latter also developing a Linux-based mobile operating system like Google. In September 2007, InformationWeek covered an Evalueserve study reporting

that Google had filed several patent applications in the area of mobile telephony.

Since 2008, Android has seen numerous updates 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, with the first few Android versions being called "Cupcake", "Donut", "Eclair", and "Froyo", in that order. During its announcement of Android KitKat in 2013, Google explained that "Since these devices make our lives so sweet, each Android version is named after a dessert", although a Google spokesperson told CNN in an interview that "It's kind of like an internal team thing, and we prefer to be a little bit — how should I say — a bit inscrutable in the matter, I'll say". In 2010, Google launched its Nexus series of devices, a lineup in which Google partnered with different device manufacturers to produce new devices and introduce new Android versions. The series was described as having "played a pivotal role in Android's history by introducing new software iterations and hardware standards across the board", and became known for its "bloat-free" software with "timely ... updates". At its developer conference in May 2013, Google announced a special version of the Samsung Galaxy S4, where, instead of using Samsung's own Android customization, the phone ran "stock Android" and was promised to receive new system updates fast. The device would become the start of the Google Play edition program, and was followed by other devices, including the HTC One Google Play edition, and Moto G

Google Play edition. In 2015, Ars Technica wrote that "Earlier this week, the last of the Google Play edition Android phones in Google's online storefront were listed as "no longer available for sale" and that "Now they're all gone, and it looks a whole lot like the program has wrapped up".

1.2.2 HOW TO IMPLEMENT

Android is developed by Google until the latest changes and updates are ready to be released, at which point the source code is made available to the Android Open Source Project (AOSP), an open source initiative led by Google. The AOSP code can be found without modification on select devices, mainly the Nexus and Pixel series of devices. The source code is, in turn, customized and adapted by original equipment manufacturers (OEMs) to run on their hardware.

1. UPDATE SCHEDULE: Google announces major incremental upgrades to Android on a yearly basis. The updates can be installed on devices over-the-air. The latest major release is 8.0 "Oreo", announced in March 2017, and released the following August.

The extensive variation of hardware in Android devices causes significant delays for software upgrades, with new versions of the operating system and security patches typically taking months before reaching consumers, or sometimes not at all. In 2012, Google began decoupling certain aspects of the operating system (particularly its core applications) so they could be updated through the Google Play store independently of the OS. One of those

components, Google Play Services, is a closed-source system-level process providing APIs for Google services, installed automatically on nearly all devices running Android 2.2 "Froyo" and higher.

With these changes, Google can add new system functionality through Play Services and update apps without having to distribute an upgrade to the operating system itself. In May 2017, with the announcement of Android 8.0, Google introduced Project Treble, a major re-architect of the Android OS framework designed to make it easier, faster, and less costly for manufacturers to update devices to newer versions of Android. Project Treble separates the vendor implementation (device-specific, lower-level software written by silicon manufacturers) from the Android OS framework via a new "vendor interface". In September 2017, Google's Project Treble team revealed that, as part of their efforts to improve the security lifecycle of Android devices, Google had managed to get the Linux Foundation to agree to extend the support lifecycle of the Linux Long-Term Support (LTS) kernel branch from the 2 years that it has historically lasted to 6 years for future versions of the LTS kernel, starting with Linux kernel 4.4.

2. LINUX KERNEL—Android's kernel is based on one of the Linux kernel's longterm support (LTS) branches. As of 2017, Android devices mainly use versions 3.18 or 4.4 of the Linux kernel. The actual kernel depends on the individual device.

Android's variant of the Linux kernel has further architectural changes that are implemented by Google outside the typical Linux kernel development cycle, such as the inclusion of components like device trees, ashmem, ION, and different out of memory (OOM) handling.

In December 2011, Greg Kroah-Hartman announced the start of Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. Linux included the auto sleep and wake locks capabilities in the 3.5 kernel, after many previous attempts at merger. The interfaces are the same but the upstream Linux implementation allows for two different suspend modes: to memory (the traditional suspend that Android uses), and to disk (hibernate, as it is known on the desktop). Google maintains a public code repository that contains their experimental work to re-base Android off the latest stable Linux versions. The flash storage on Android devices is split into several partitions, such as /system for the operating system itself, and /data for user data and application installations. In contrast to desktop Linux distributions, Android device owners are not given root access to the operating system and sensitive partitions such as /system are read-only. However, root access can be obtained by exploiting security flaws in Android, which is used frequently by the open-

source community to enhance the capabilities of their devices, but also by malicious parties to install viruses and malware.

3.SOFTWARE STACK: On top of the Linux kernel, there are the middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries. Development of the Linux kernel continues independently of other Android's source code bases.

Until version 5.0, Android used Dalvik as a process virtual machine with tracebased just-in-time (JIT) compilation to run Dalvik "dex-code" (Dalvik Executable), which is usually translated from the Java bytecode. Following the trace-based JIT principle, in addition to interpreting the majority of application code, Dalvik performs the compilation and native execution of select frequently executed code segments ("traces") each time an application is launched. Android 4.4 introduced Android Runtime (ART) as a new runtime environment, which uses ahead-of-time (AOT) compilation to entirely compile the application bytecode into machine code upon the installation of an application. In Android 4.4, ART was an experimental feature and not enabled by default; it became the only runtime option in the next major version of Android, 5.0. For its Java library, the Android platform uses a subset of the now discontinued Apache Harmony project. In December 2015, Google announced that the next version of Android would switch to a Java implementation based on OpenJDK. Android's standard C library, Bionic,

was developed by Google specifically for Android, as a derivation of the BSD's standard C library code.

Android has another operating system, Trusty OS, within it, as a part of "Trusty" "software components supporting a Trusted Execution Environment (TEE) on mobile devices." "Trusty and the Trusty API are subject to change. Applications for the Trusty OS can be written in C/C++ (C++ support is limited), and they have access to a small C library. All Trusty applications are single-threaded; multithreading in Trusty userspace currently is unsupported.

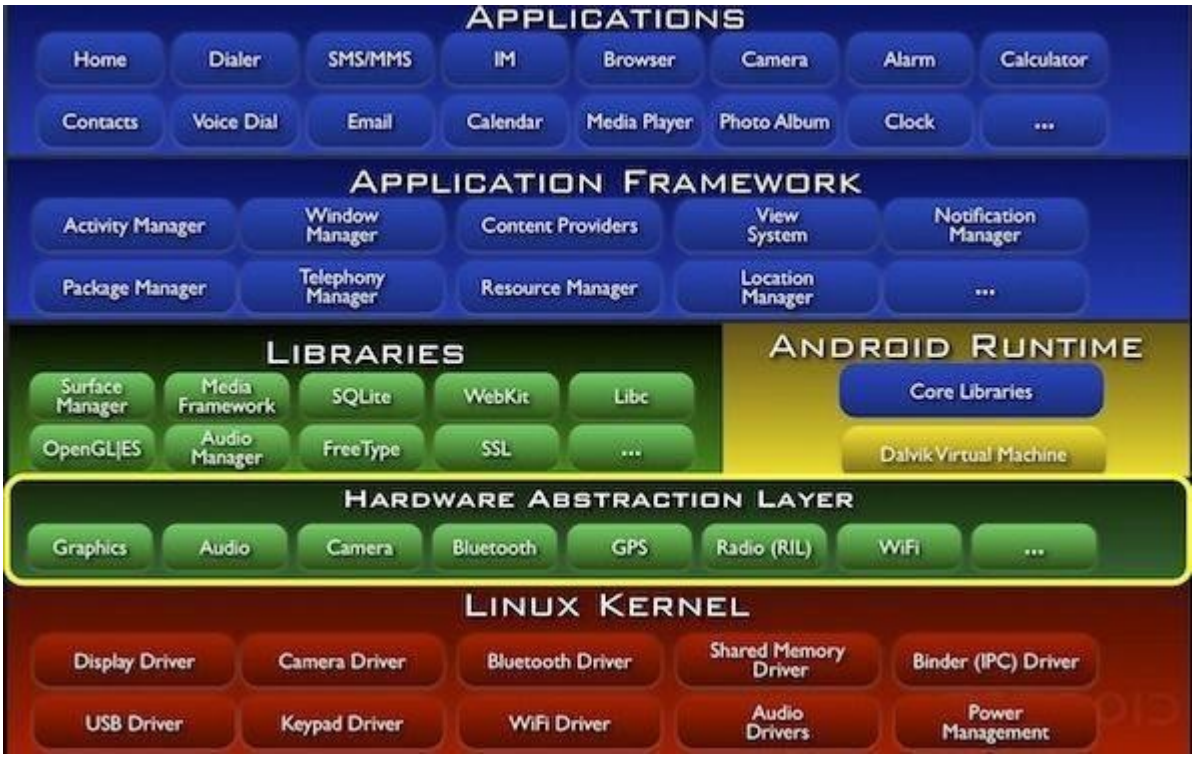

Fig 4.1 Android's Architecture Diagram

4.OPEN-SOURCE COMMUNITY: Android's source code is released by Google under an open source license, and its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, with a comparable level of quality; 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.

TECHNICAL SECURITY FEATURES

Android applications run in a sandbox, 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, however this may not be possible for pre-installed apps. It is not possible, for example, to turn off the microphone access of the pre-installed camera app without disabling the camera completely. This is valid also in Android versions 7 and 8. Before installing an application, the Google Play store displays a list of the requirements an app needs to function. After reviewing these permissions, the user can choose to accept or refuse them, installing the application only if they accept.

In Android 6.0 "Marshmallow", the permissions system was changed; apps are no longer automatically granted all of their specified permissions at installation time. An opt-in system is used instead, in which users are prompted to grant or deny individual permissions to an app when they are needed for the first time. Applications remember the grants, which can be revoked by the user at any time. The new permissions model is used only by applications developed for Marshmallow using its software development kit (SDK), and older apps will continue to use the previous all-or-nothing approach. Permissions can still be revoked for those apps, though this might prevent them from working properly, and a warning is displayed to that effect.

1.2.3 SOFTWARE ENVIRONMENT

Android software development is the process by which new applications are created for devices running the Android operating system. Officially, apps can be written using Java, C++ or Kotlin using the Android software development kit (SDK). Third party tools, development environments and language support have also continued to evolve and expand since the initial SDK was released in 2008.

ANDROID SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely). Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older

versions of the Android platform in case developers wish to target their applications at older devices.

Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under "/data/app" folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

DEBUG BRIDGE ANDROID

The Android Debug Bridge (ADB) is a toolkit included in the Android SDK package. It consists of both client and server-side programs that communicate with one another. The ADB is typically accessed through the command-line interface, although numerous graphical user interfaces exist to control ADB.

The format for issuing commands through the ADB is typically:

adb [-d|-e|-s <serial Number>] <command>

where –d is the option for specifying the USB-attached device,

-e is for indication a running Android emulator on the computer,

-s is for specifying either one by its adb-assigned serial number.

If there is only one attached device or running emulator, these options are not necessary.

ANDROID NDK

Libraries written in C/C++ can be compiled to ARM, MIPS or x86 native code (or their 64bit variants) and installed using the Android Native Development Kit (NDK). These native libraries can be called from Java code running under the Dalvik VM using the System, load Library call, which is part of the standard Android Java classes.

Complete applications can be compiled and installed using traditional development tools. However, according to the Android documentation, NDK should not be used solely because the developer prefers to program in C/C++, as using NDK increases complexity while most applications would not benefit from using it.

The ADB Debugger gives a root shell under the Android Emulator which allows ARM, MIPS or x86 native codes to be uploaded and executed. Native code can be compiled using Clang or GCC on a standard PC. Running native code is complicated by Android's use of a non-standard C library (libc, known as Bionic).

The graphics library that Android uses to arbitrate and control access to this device is called the Skia Graphics Library (SGL), and it has been released under an open source licence. Skia has backends for both Win32 and Unix, allowing the development of cross-platform applications, and it is the graphics

engine underlying the Google Chrome web browser. Skia is not an NDK API, though, and NDK developers use OpenGL.

It is possible to use the Android Studio with Gradle to develop NDK projects.

1.2.4 ADVANTAGES OF ANDROID

1. Supports 2D, 3D graphics

It supports various platforms like 2D and 3D. Earlier we used to watch movies and play games in almost in 2D, but nowadays various applications are using 3D format. To provide different graphics in videos, games OS should support 3D format. Android supports 2D And 3D format to provide a better advantage in videos and in games.

2. Supports Multiple Languages

Android supports different languages. We can say all famous languages about more than 100. By using this feature it is easy to adopt to different languages. Earlier in the feature phones English is to be the only language in the mobile devices.

3. Java Support

The Java supporting feature enables developers to enhance more features. As it supports Java, operating can be developed.

4. Faster Web Browser

As it enabled with web browser we surf web easily without complexity just like in a computer. It easily loads multimedia so that it makes web browsing faster.

5. It Supports MP4, 3GP, MPEG4, MIDI

It supports different types of formats. There is no need to convert from one format to another, as it enabled with different formats of audio and video styles.

6. Additional Hardware Support

Any hardware can be easily connected with the Android based devices easily. We can make a device to connect internally to get more features.

7. Video Calling

Faster data connection enables to do video call. We can take advantage of bandwidth and new generation networks using Android.

8. Open Source Framework

It makes users to make their own applications and to make changes required for themselves. Enthusiasts can make Andriod more powerful and useful by developing themselves. As it is an open source operating system, we can use it easily and without cost in the equipments.

9. Uses of Tools are Very Simple

It makes use of a single button to do more than assigned work. For example volume control button can be made to click a photo by changing simple algorithm in the android.

10. Availability of Apps

Anyone can make use lot of free apps in the app store and from other android stores. It gives freedom to install from third party users.

1.3 About Language Used

In the android there are various languages are used according to their need in the project. Mainly some types of languages are used in android development that is given below

Java – Java is the official language of Android development and is supported by Android Studio. It has a steep learning curve however. Kotlin – Kotlin was recently introduced as a secondary 'official' Java language. It is similar to Java in many ways but is a little easier to get to grips with. C/C++ – Android Studio also supports C++ with the use of the Java NDK. This allows for native coding applications, which can be handy for things like games. C++ is more complicated still however. C# – C# is a slightly more beginner friendly alternative to C or C++ that obfuscates more code. It's supported by some very handy tools like Unity and Xamarin which are great for game development and for cross platform development.

BASIC – A bonus option is to learn BASIC and try the B4A IDE from Anywhere Software. This is an easy but powerful tool, though definitely much more ‘niche’!

Corona/LUA – Another cross-platform tool build on LUA. It massively simplifies the app-building process and allows you to call native libraries. PhoneGap (HTML, CSS, JavaScript) – If you already know how to build interactive web pages, then you can use this knowledge with PhoneGap to build a more basic cross-platform app

1.3.1 JAVA LANGUAGES

Java is a general-purpose computer-programming language that is concurrent, classbased, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine(JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by 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. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath(standard libraries), and IcedTea-Web (browser plugin for applets). The latest version is Java 10, released on March 20, 2018, which follows Java 9 after only six months in line with the new release schedule. Java 8 is still supported but there will be no more security updates for Java 9. Versions earlier than Java 8 are supported by companies on a commercial basis; e.g. by Oracle back to Java 6 as of October 2017 (while they still "highly recommend that you uninstall" pre-Java 8 from at least Windows computers). Java Language is used for the backend language in the android. It validates the all the pallete of the android.

1.3.2 XML Language

XML language is used in the front end of the android language. And it is responsible for the designing of the android application. XML stands for extensible mark-up language. In computing, Extensible Markup Language (XML) is a mark-up language that defines a set of rules for encoding documents in a format that is both human-readable and machinereadable. The

W3C's XML 1.0 Specification and several other related specifications all of them free open standards—define XML. 14)

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services. Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data. XML language is implemented only on the browsers. It is the strict name resolution that means in this application is the strict typed name. In this language we created our own tags.

CHAPTER 2 :

SOFTWARE DEVELOPMENT LIFE CYCLE

2.1 REQUIREMENT ANALYSIS PHASE

The Requirements Analysis Phase begins when the previous phase objectives have been achieved. Documentation related to user requirements from the Concept Development Phase and the Planning Phase shall be used as the basis for further user needs analysis and the development of detailed requirements. Multiple-release projects require only one iteration of the Requirements Analysis Phase, which should involve requirements definition for all planned releases.

The objective of this phase is to define in more detail the system inputs, processes, outputs and interfaces. At the end of this phase the system's processes will be defined at the functional level, meaning the functions to be performed will be known, but not necessarily how they will be performed. Unless specifically constrained by the Project Charter, Requirements Analysis should not consider the computer programs, files and data streams.

Requirements Analysis will identify and consider the risks related to how the technology will be integrated into the standard operating procedures.

Requirements Analysis will collect the functional and system requirements of the business process, the user requirements and the operational requirements (e.g., when operational what is necessary to keep the system up and running).

2.1.1 SYSTEM REQUIREMENT SPECIFICATION

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure. The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

Purpose

The purpose of this document is to give a detailed description of the requirements for this application. It will illustrate the purpose and complete declaration for the development of system. It will also explain system

constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

2.1.2 HARDWARE & SOFTWARE REQUIREMENT

Hardware Specification:

- Hard Disk – 5 GB
- A computer with at least 2GB of RAM(Random Access Memory)
- Processor – i3

Software Requirements:

- Android Studio
- Languages: Java and XML

2.1.3 FUNCTIONAL REQUIREMENT

In Software engineering and systems engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are

supported by non-functional requirements which impose constraints on the design or implementation.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

2.1.4 NON –FUNCTIONAL REQUIREMENT

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements.

Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of, an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of, an

overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed.

Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioural requirements". Informally these are sometimes called the "ileitis", from attributes like stability and portability. Qualities—that is non-functional requirements—can be divided into two main categories:

1. Execution qualities, such as safety, security and usability, which are observable during operation. 2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

2.1.5 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

2.1.6 TECHNICAL FEASIBILITY

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

Does the existing technology sufficient for the suggested one?

Can the system expand if developed?

2.1.7 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

The costs conduct a full system investigation.

The cost of the hardware and software.

The benefits in the form of reduced costs or fewer costly errors. Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.1.8 BEHAVIORAL FEASIBILITY

This includes the following questions:

Is there sufficient support for the users?

Will the proposed system cause harm? The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.2 DESIGN PHASE

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

2.2.1 LOGICAL DESIGN:

The logical flow of a system and define the boundaries of a system. It includes the following steps:

Reviews the current physical system – its data flows, file content, volumes, Frequencies etc.

Prepares output specifications – that is, determines the format, content and Frequency of reports. 21)

Prepares input specifications – format, content and most of the input functions.

Prepares edit, security and control specifications.

Specifies the implementation plan.

Prepares a logical design walk through of the information flow, output, input, Controls and implementation plan.

Reviews benefits, costs, target dates and system constraints.

2.2.2 PHYSICAL DESIGN:

Physical system produces the working systems by define the design specifications that tell the programmers exactly what the candidate system must do. It includes the following steps.

Design the physical system.

Specify input and output media.

Design the database and specify backup procedures.

Design physical information flow through the system and a physical design.
Plan system implementation.

Prepare a conversion schedule and target date.

Determine training procedures, courses and timetable.

Devise a test and implementation plan and specify any new hardware/software.

Update benefits , costs , conversion date and system constraints

2.2.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

DATA FLOW DIAGRAM