SMART VIDEO SURVEILLANCE

A PROJECT REPORT

Submitted by

KHOT HARISH SHRIDHAR	**B80784231**
KHATAL SONALI BHANUDAS	**B80784229**
GOTE SWATI RAMDAS	**B80784224**
PANDARGE SANGMESH SHIVKANT	**B80784244**

In partial fulfillment for the award of the degree

of

BACHLOR OF ENGINEERING

IN

COMPUTER ENGINEERING

Guided by

Danny J. Pereira

GOVERNMENT COLLEGE OF ENGINEERING AND RESEARCH, AWASARI. DIST-PUNE

PUNE UNIVERSITY: PUNE 411 007

APRIL-MAY 2015

CERTIFICATE

This is to certify that the dissertation entitled "**Smart Video Surveillance**" submitted by

KHOT HARISH SHRIDHAR	B80784231
KHATAL SONALI BHANUDAS	B80784229
GOTE SWATI RAMDAS	B80784224
PANDARGE SANGMESH SHIVKANT	B80784244

is a bona-fide work carried out under supervision and guidance of Mr. Danny J. Pereira and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University for award of Bachelor of Engineering (Computer Engineering) Degree. This project work is not been earlier submitted to any other institute or University for the award of the degree.

Place :

Date :

Mr. Danny J. Pereira
Guide

Mr. Danny J. Pereira
Head of the Department

External Examiner

Prof. S.V. Joshi
Principal
Government College of Engineering & Research Awasari

ACKNOWLEDGEMENT

We take this opportunity to thank our project guide **Mr. D. J. Pereira** and Head of the Department **Mr. D. J. Pereira** for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of the Department of Computer of Government College of Engineering and Research, Awasari for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, Internet access and important books.

ABSTRACT

This project has a special feature of smart video transfer and capture feature. Smart Video Surveillance is capable of enhancing situational awareness across multiple scales of space and time. This project de-scribes mobile based remote control and surveillance architecture. It is very suitable for remote bank monitoring, shop etc. The idea is to set up a mobile with a GSM Modem at banks, shops which can be used to transmit/receive video/photos and/or commands to and from the administrator/owner. Once the comparison is done and an intrusion is found, it sends the streamed video from server to remote administrator over android phone, laptop & pc. Admin can then take appropriate action and alert local security.

CONTENTS

List of Figures

List of Abbreviation

Sr no	Abbreviation	Meaning
1	WWW	World Wide Web
2	HTML	Hypertext Markup Language
3	IDE	Integrated Development Environment
4	HTTP	Hypertext Transfer Protocol
5	SMTP	Simple Mail Transport Protocol
6	API	Application Program Interface
7	GUI	Graphical User Interface
8	XML	Extensible Markup Language
9	SOAP	Simple Object Access Protocol
10	SDK	Software Development Kit
11	JDK	Java Development Kit
12	JNI	Java Native Interface

Chapter 1

INTRODUCTION

1.1 Introduction

Observing or analyzing a particular site for safety and business purposes is known as video surveillance. Security and crime control concerns are the motivating factors for the deployment of video surveillance cameras. Video surveillance cameras are used in shopping centers, public places, banking institutions, companies and ATM machines.

Nowadays, researches experience continuous growth in network surveillance. The reason being is the instability incidents that are happening all around the world. Therefore, there is a need of a smart surveillance system for intelligent monitoring that captures data in real time, transmits, processes and understands the information related to those monitored. The video data can be used as a forensic tool for after-crime inspection. Hence, these systems ensure high level of security at public places which is usually an extremely complex challenge. As video cameras are available at good price in the market, hence video surveillance systems have become more popular. Video surveillance systems through smart phone have wide range of applications like traffic monitoring and human activity understanding.

Benefits of Video Surveillance is:

1. **Availability**- There was a time when the surveillance techniques were utilized only in shopping centers and malls. Now-a-days, you can notice closed-circuit televisions almost at any place you visit, from a small store to homes and holy places. As a result, they guarantee greater public security at a fraction of the cost.
2. **Real-Time Monitoring**- Traditionally big organizations have always had the benefits of video surveillance manned by security professionals. In the past times, the events captured on video were used to expose important information and work as proof after the event happened. But, modern technologies let users to check and reply to alarms immediately. Using a number of video cameras, a large amount of visual data is captured that is to be monitored and screened for intrusion detection. Presently, the surveillance systems used requires constant human vigilance. However, the humans have limited abilities to perform in real-time which reduce the actual usability of such surveillance systems. Also such surveillance systems are not reliable for real time threat detection. From the perspective of forensic investigation, a large amount of video data obtained from surveillance video tapes need to be analyzed and this task is very tedious and error prone for a human investigator. To overcome this drawback, automatic video analysis system is developed that continuously monitors a given situation and reacts in real-time. The proposed system has an ability to sense intrusion and respond to it in real time.

However, the humans have limited abilities to perform in real- time which reduce the actual usability of such surveillance systems. Also such surveillance systems are not reliable for real time threat detection. From the perspective of forensic investigation, a large amount of video data obtained from surveillance video tapes need to be analyzed and this task is very tedious and error prone for a human investigator. To overcome this drawback, automatic video analysis system is developed that continuously monitors a given situation and reacts in real-time. The proposed system has an ability to sense intrusion and respond to it in real time.

1.2 PROBLEM DEFINITION

This project is aiming at detecting unusual events from surveillance videos. All the videos are shot using a single mobile camera. One example of such an event could be detection of events by a theft when a shop is closed.

Input:

1. Videos shot using mobile camera with one event per shot.

Expected Output:

1. Automatically detect the unusual activity
2. Give the message alert

Chapter 2

LITERATURE SURVEY

Video surveillance has been around from before the introduction of the VCR (Video Cassette Recorder) and actually as far back as the birth of the V-2 rocket in Germany in 1942 where it was used for observing launches. The first commercially available camera was introduced by the American company Verification in 1949 and was advertised to :keep an eye on dangerous industrial processes or bring a close-up of demonstrations and surgical operations to large groups of students The first actual surveillance cameras were installed in the late sixties, start seventies. The police department in New York installed cameras in the Municipal Building and on Time Square in order to prevent criminal activities - unfortunately the crime rates stayed at status quo. Since this was before the introduction of the VCR, the monitors had to be watched at all times.

Only after the VCR became a mass produced and adorable product, did surveillance cameras become a wide-spread phenomenon. Cam-eras was installed in major London underground station since 1975, store owners and banks installed cameras for use as evidence after a robbery or other criminal activity. However there were still some major drawbacks using VCRs and analogue technology. The tapes had to be changed or would wear out and the cameras did not perform well in low light or at night. Cameras and technology in general has kept improving with introduction of e.g. the CCD chip, digital multiplexing and motion only recording in the 1990s. After the terrorist attack on World Trade Center on September 11th 2001, more cameras were installed and more focus was put on processing the surveillance video. It was now time to use the video, in real time, to detect persons. On site of the Statue of Liberty facial recognition software was installed for terrorist attack prevention, in schools for tracking missing children and sexual o enders and in airports as verification of identity. In the last few years as computer power has increased and the internet has become accessible from almost anywhere there have been breakthroughs in video surveillance. With good compression algorithms available and increased storage space many days or weeks of video can be saved and more cameras can be connected to the same computer.

Fast development in the technology has increased the risk of intrusion. Using security cameras allows a person to monitor his property. The majority of organization and administrations are making use of such security cameras with the intention to save their business as well as property from terrorists and illegal entry. Nowadays, the security cameras have become much more advanced, reasonable, smaller and straight forward .A number of video surveillance systems has been proposed for different purposes. Drew Ostheimeretal2006 (Drew Ostheimeretal2006) proposed an automated and distributed real-time video surveillance system which can be used for the detection of objects and events in a wide range of applications. The system captures video from multiple sources which is then processed and streamed over the internet for viewing and analysis. The proposed system is flexible as the components of the system can be interconnected in several manners. The experimental results of the system show that it can handle multiple video data running on standard computers and yielding uid video. A number of interconnected clients can view the multiple video feeds simultaneously. Alberto Amato (Alberto Amatole al2005) proposed a semantic event detection system based on a neural classifier that screens continuous video streams and detect relevant events for video surveillance. The goal of the system an easy way

to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it is to improve the aware-ness of security personal and decision makers by collecting real-time information automatically. The system raises an alarm whenever un-acceptable movements are detected. Hence, the system has the ability to detect mobile objects in the scene and to classify their movements (as allowed or disallowed). Wann-Yun Shieh(Wann-Yun Shiehet al., 2009) proposed a human-shape-based falling algorithm and this algorithm was implemented in a multi-camera video surveillance system. The algorithm is implemented in real world environment for functionality proof. In this algorithm, multiple cameras are used to fetch the images from different regions required to monitor. A falling pattern recognition approach is used to determine if an accidental falling has occurred. Also, in that case a short message will be sent to someone who needs to be alerted. Hae-Min Moon(Hae-Min Moonet al., 2010) proposed the system on human identification method that uses height and clothing-colour information appropriate for the intelligent video surveillance system based on smart card. Reliable feature information can be obtained using the smartcard. It uses octree-based colour quantization technique to the clothing region for colour extraction and height is extracted from the geometrical information of the images.

Chapter 3

SOFTWARE REQUIREMENT & SPECIFICATION

3.1 Project scope

Our system allows user to view videos even if he is at some remote place. The system provides the functionality of online video streaming so that user can view the videos from web browser. Our system pro-vides a software solution for image matching and intrusion detection. We do not require use of any additional hardware for this purpose. Our system uses image matching technique, so it gives more precise and ac-curate results. Entire Smart surveillance can be made remote using this architecture. User can even control the system through a remote place. He can give commands to switch on/o the system. The system provides real-time monitoring. The user is notified as soon as the intrusion is detected. Thus, the user can take appropriate action without any delay. Surveillance is integrated with intelligent video movement detection analysis systems combine with SMS, email alarm notification system.

3.2 User classes and characteristics

The user of this product/system will be any person which wants to secure home/home/shop etc. When he/she is not present at that movement.

3.3 Operating Environment

Software has two major component one the server and the second one is the mobile application. The server will required Windows XP/Vista/7 machine with minimum 1GB RAM and 100 GB hard disk. The server machine also required WI-FI devices sing which it can create Wireless Ad-hoc network. Mobile application will support Android phones so at least 2 Android devices required getting the output.

3.4 Assumptions and Dependencies

The user is expected to have Android Mobile phones and should be able to send and receive data when connected to Wi-Fi range.

3.5 System Features

Server:

1. System should support Android handset
2. System should prove web based access (HTTP support)
3. System should have built in camera on mobile handset
4. System should capable to detect any movement happen
5. System should allow capturing video when any movement happen
6. System should send message to another system(mobile)
7. System should allow user to watch live video or movements

Mobile Client:

1. System should prove web based access (HTTP support)
2. System should allow getting message from another mobile.
3. System should allow users to watch live movement.

3.6 External Interface Requirements

3.6.1 User Interfaces

Server Side:

1. Camera Open Interface

Client Side:

1. Welcome screen
2. Camera Open Interface
3. Message form

3.6.1 Hardware Interfaces

Mobile application will get installed on mobile devices. These mobile devices should have WIFI device thorough which it will connect to server.

3.6.2 Software Interfaces

1. Operating System: Windows XP/Windows Vista/Windows 7.
2. Database: SQLite
3. Android 2.2 supported mobile handset

3.6.3 Communication Interfaces

Here we will be using WI-FI network and going to create our own communication protocol. Software will also support BASE64 encryption logic while sending data to server. Server will support HTTP protocol for web based access.

3.7 Nonfunctional Requirements

3.7.1 Performance Requirements

For good performance the server should be tuned to server only server process and most of the RAM should be used for our application. Mo-bile application should use as much possible RAM. KVM should be tuned on mobile to provide extra address space to application

3.7.2 Security Requirements

All the data will be shared using peer to peer network. And this app is used for security (house/shop) purpose. So there is no such issue of safety and security of data.

3.8 ANALYSIS MODELS

3.8.1 Data flow diagrams

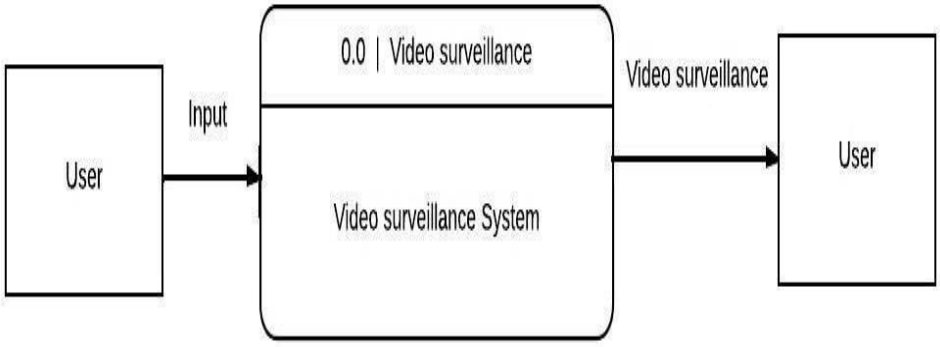


Fig.3.8.1 Data flow diagrams-0

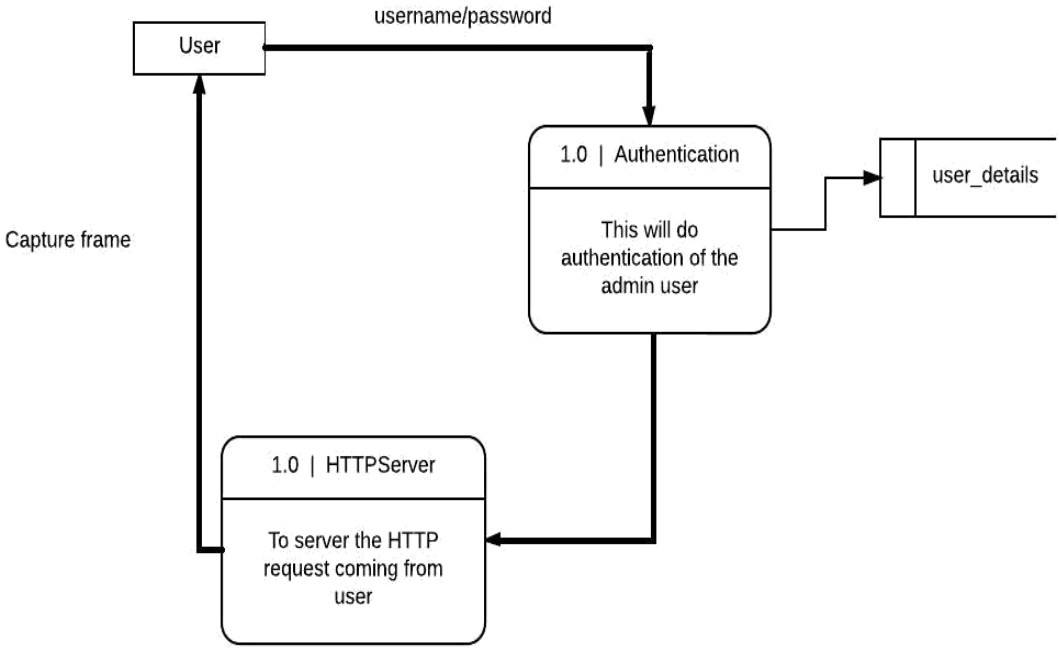


Fig.3.8.2 Data flow diagram-1

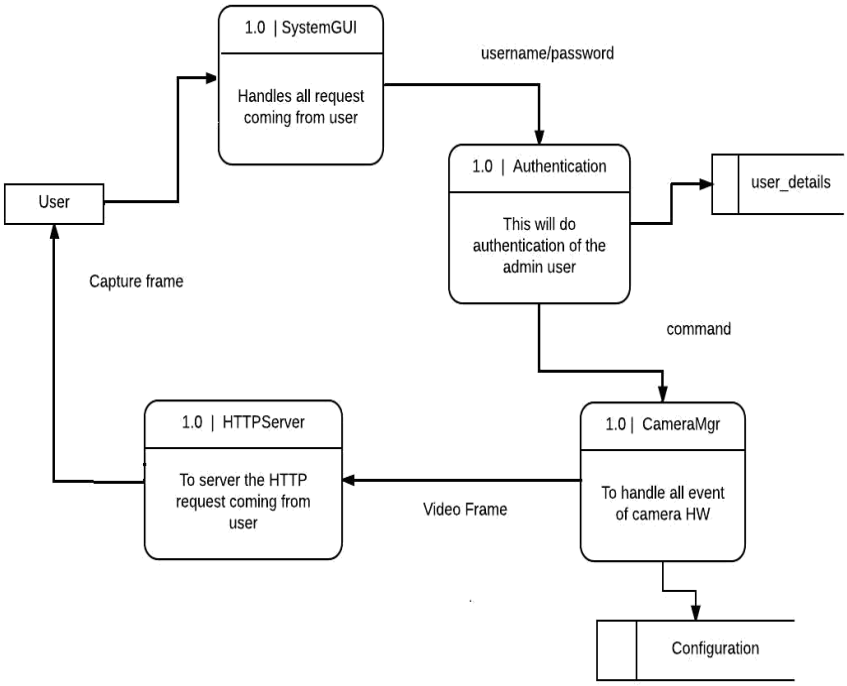


Fig.3.8.3 Data flow diagram-2

3.8.2 Class Diagram

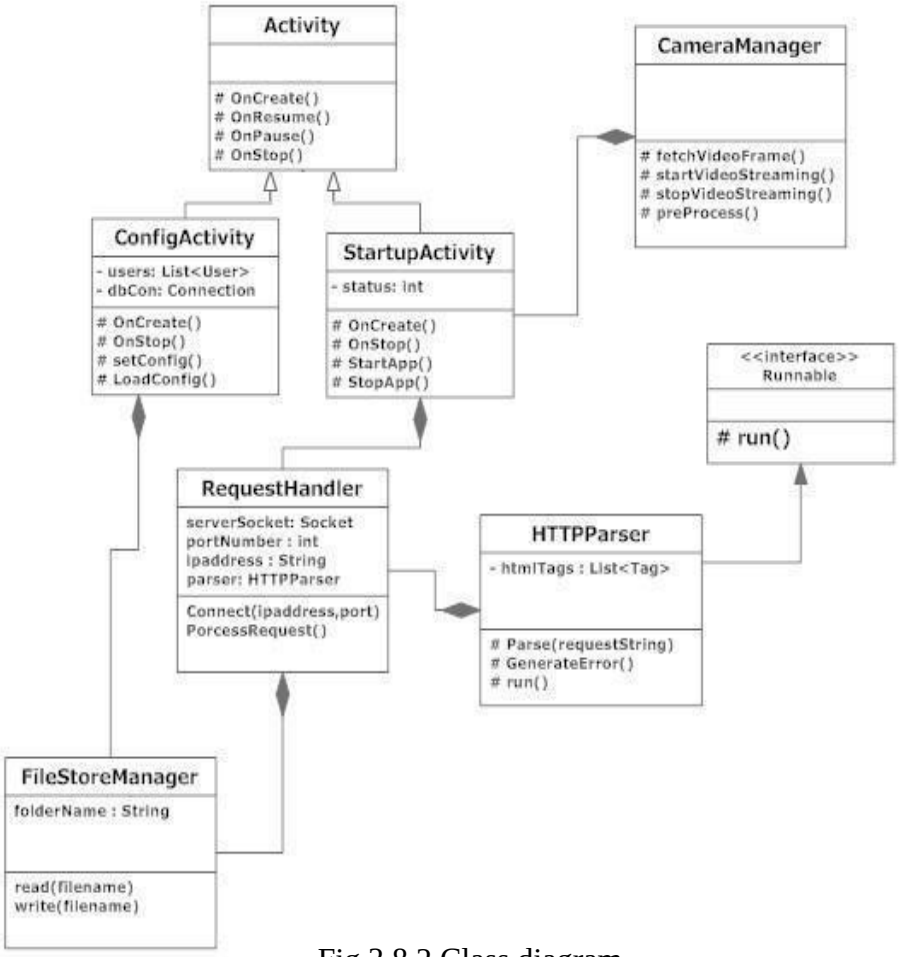


Fig.3.8.2 Class diagram

Chapter 4

SYSTEM DESIGN

4.1 System Architecture

Video surveillance is a peer-to-peer (P2P) based application . The main goal of this application is to transfer a message to mobile phone when any movement is detected Video Surveillance Technology is used with the help of Camera of mobile, If mobile camera detects any movement then it automatically send message to another mobile.

Web services: For the connection between the app and server the HTTP protocol will be used. For every access of the data the web services will be needed.

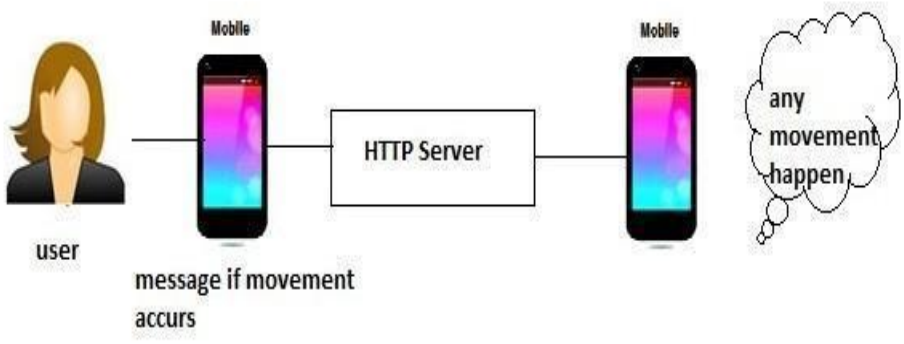


Fig 4.1.1: System Architecture

Server: Android mobile is keep in shop/house etc. This mobile continuously checking for movement. If any movement is detected then server side mobile send a alert message to another mobile.

User: User get a message alert from server side mobile ,if any movement is detected Video surveillance provides a cost actives alternative for public safety workers to monitor activities in almost any location, without adding more feet on the street. Both xed and mobile video services can be deployed to deliver a range of benefits to communities while increasing the efficiency and effectiveness of public safety workers. The following diagram shows the flow of our system and the processes involved.

The administrator starts the video surveillance system. As soon as the surveillance system is initialized, the system checks if the web camera is connected or not. If the web camera is not connected to the system then it will display an error message. Otherwise, the sys-tem continuously starts capturing images. A standard image is already stored in a separate le. The captured images are continuously compared with this standard image and are checked for any intrusion. In case of intrusion, a SMS will be sent to the administrator/owner for appropriate action to be taken. User can then login to the surveillance web application to view the most recent videos.

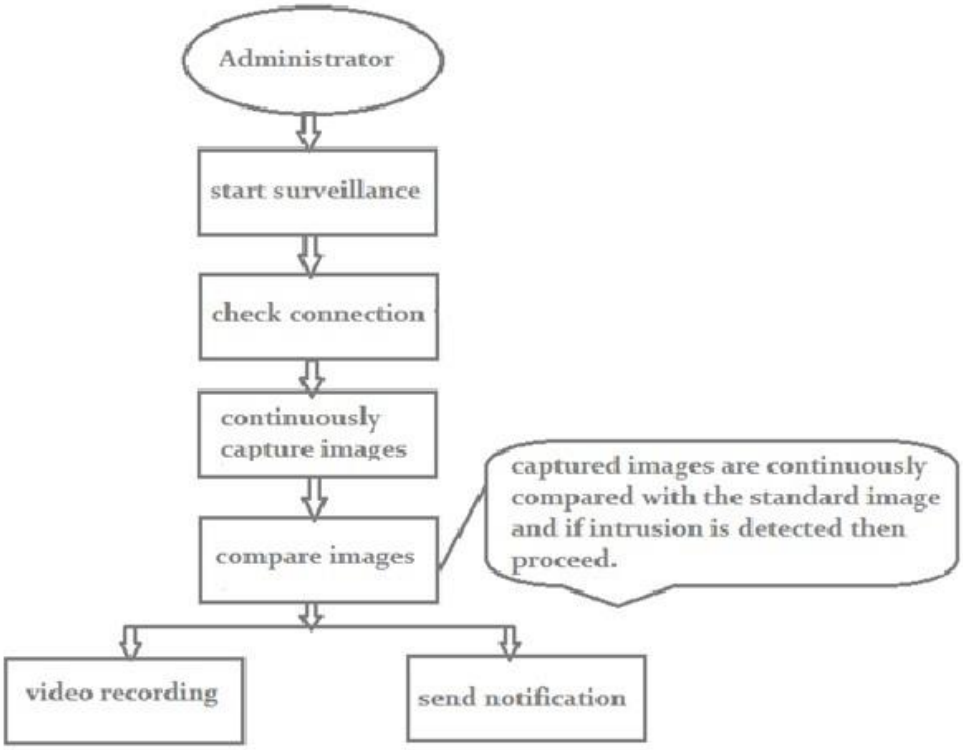


Fig 4.1.2: Flowchart of proposed system

The system waits for a specified amount of time for response commands (SMS) from any of the owners, after which it takes necessary action itself. E.g. the device starts alarming. It can store mobile numbers for all the administrators/owners who need to be contacted in case of emergency. The system keeps track/log of all the activities. Hence detailed record of messages sent and received is maintained. Administrator can send commands to control switch on/o of the device. User can also send a series of command sequences scheduled for a later time the commands will be executed automatically at the server when the time arrive System receives commands from administrators which are then used to take necessary actions. The commands may include activating/deactivating a rely , setting etc. The system only responds to owners mobile numbers SMS received from any other mobiles will be rejected. Moreover the communication via SMS is password protected. Hence any other user cannot control the system from one of the owner's mobile number. The entire smart surveillance is made remote using this architecture Benefits of Proposed System:

1. Our system allows user to view videos even if he is at some re-mote place. The system provides the functionality of online video streaming so that user can view the videos from web browser.
2. Our system provides a software solution for image matching and intrusion detection. We do not require use of any additional hard-ware for this purpose.
3. Our system uses image matching technique, so it gives more precise and accurate results.
4. Entire Smart surveillance can be made remote using this architecture. User can even control the system through a remote place. He can give commands to switch on/o the system.
5. The system provides real time monitoring. The user is noticed as soon as the intrusion is detected. Thus, the user can take appropriate action without any delay.
6. Surveillance is integrated with intelligent video movement detection analysis systems combine with SMS, email alarm notification system

4.2 UML Diagrams

4.2.1 Use case Diagram

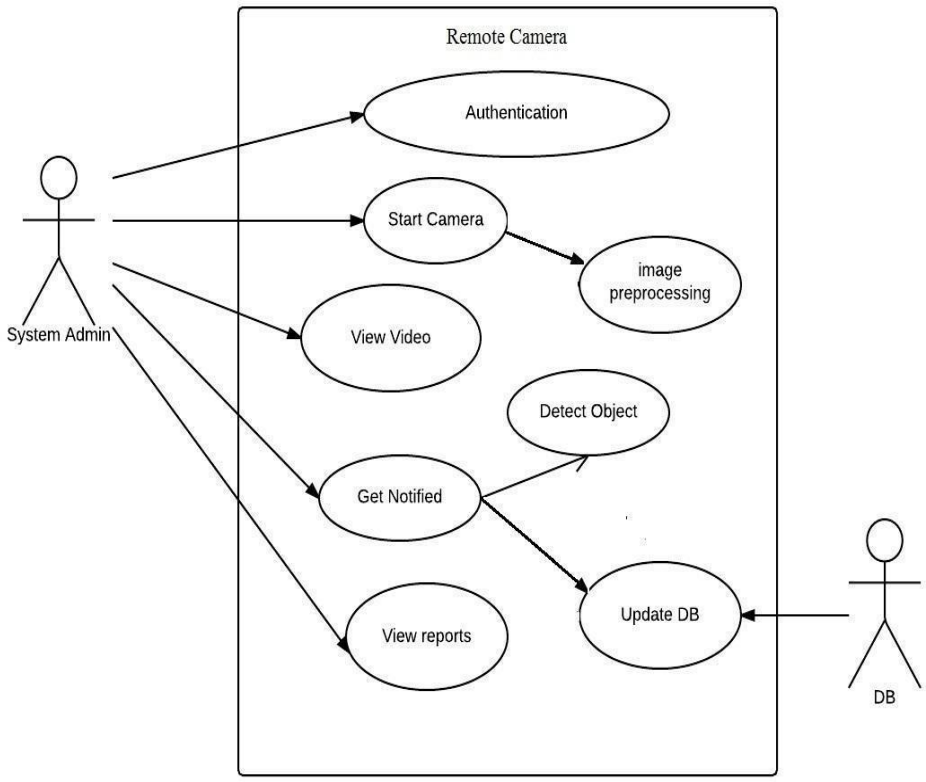


Fig 4.2.1: Use case diagram

4.2.2 Class Diagram

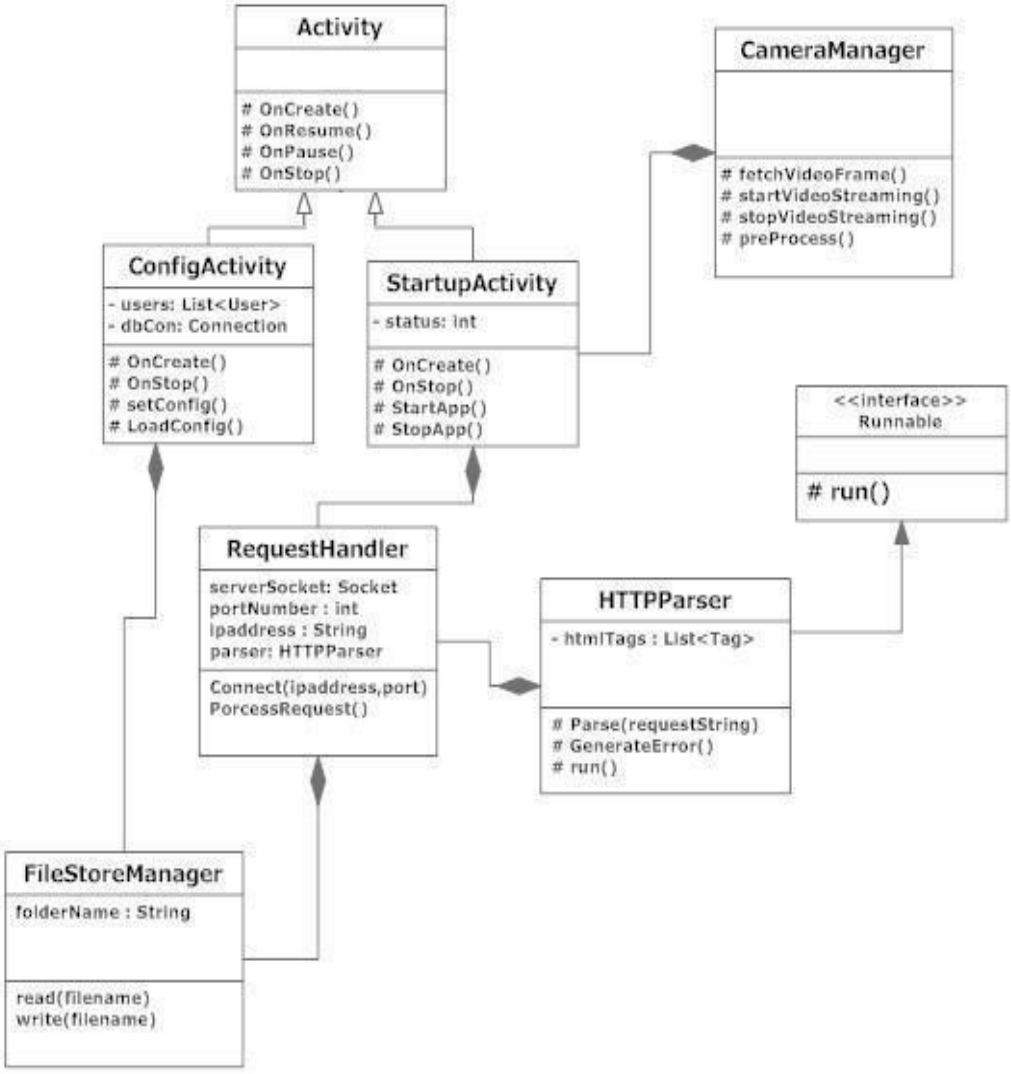


Fig 4.2.2: Class diagram

4.2.3 Component Diagram

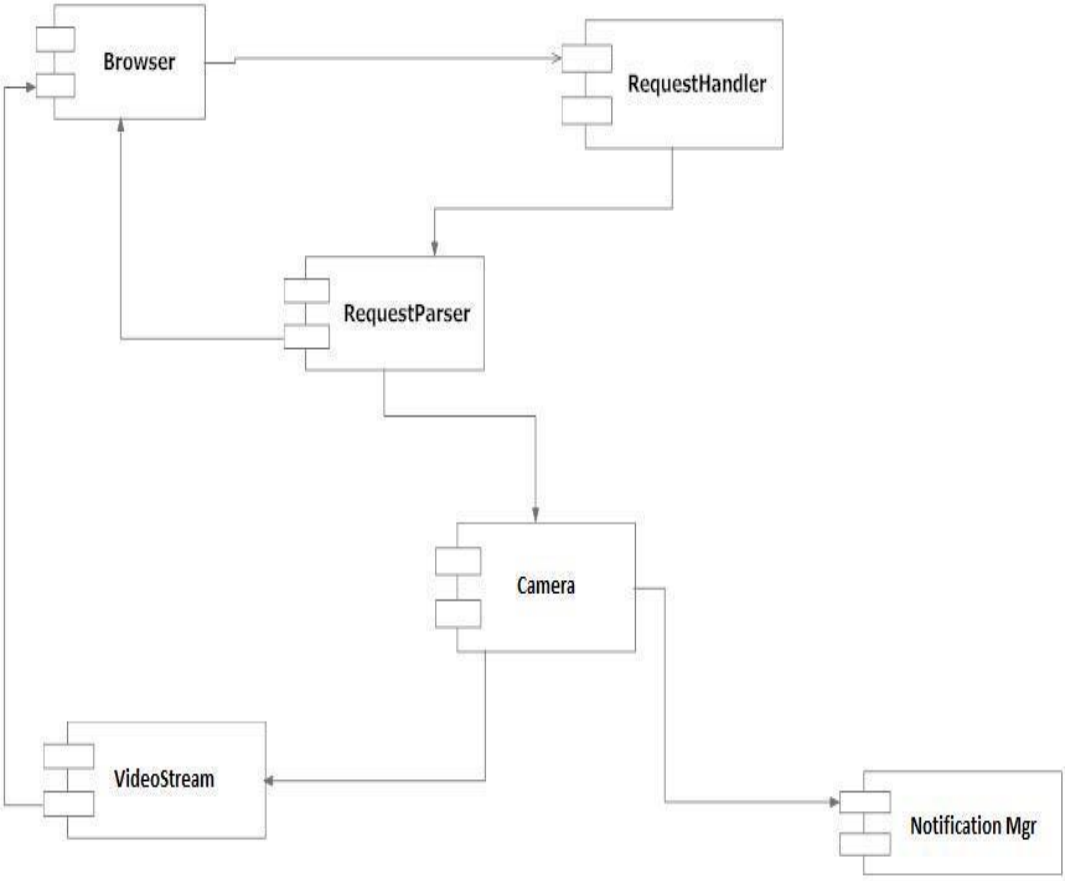


Fig 4.2.3: Component diagram

4.2.4 Activity Diagram

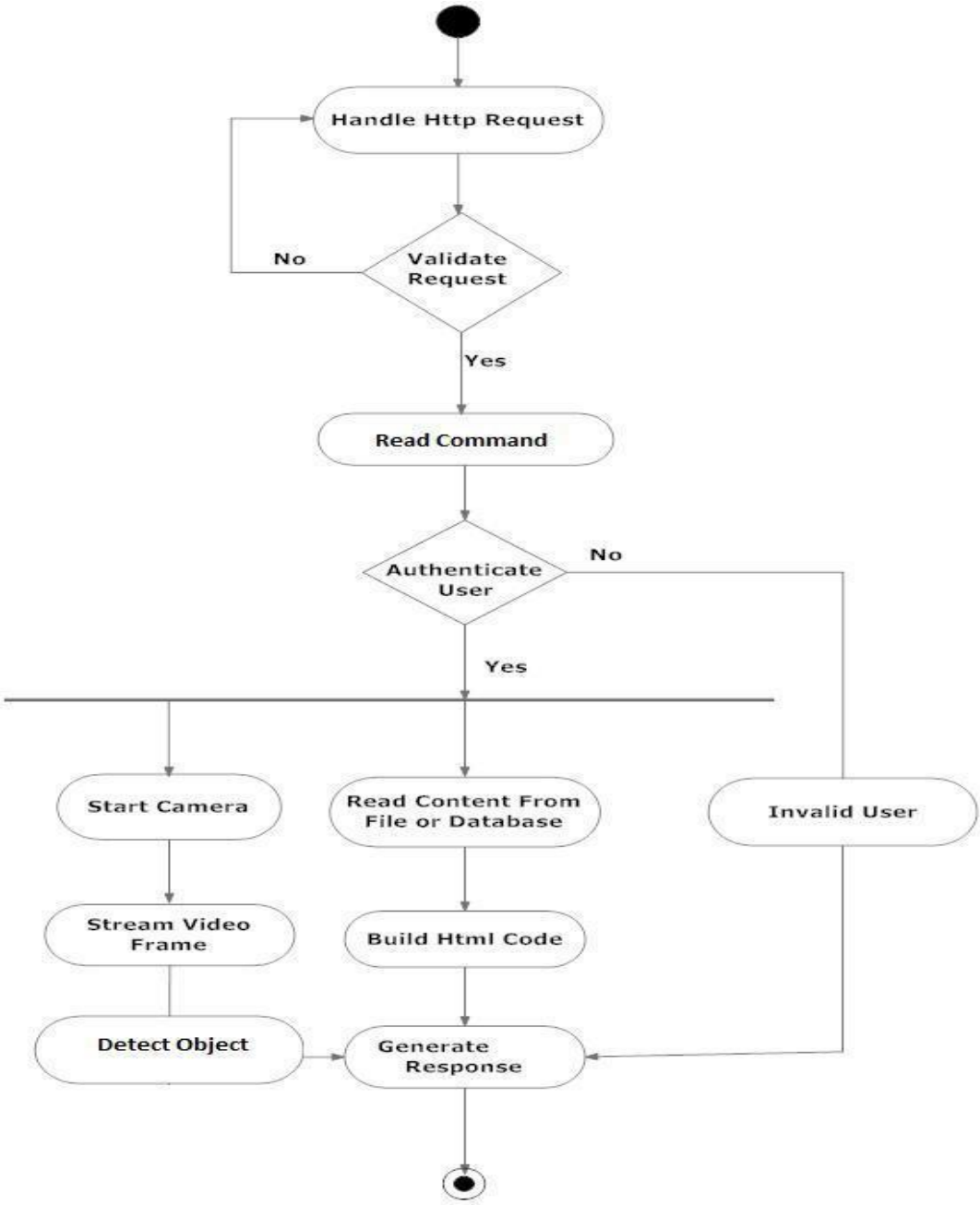


Fig 4.2.4: Activity Diagram

4.2.5 Deployment Diagram

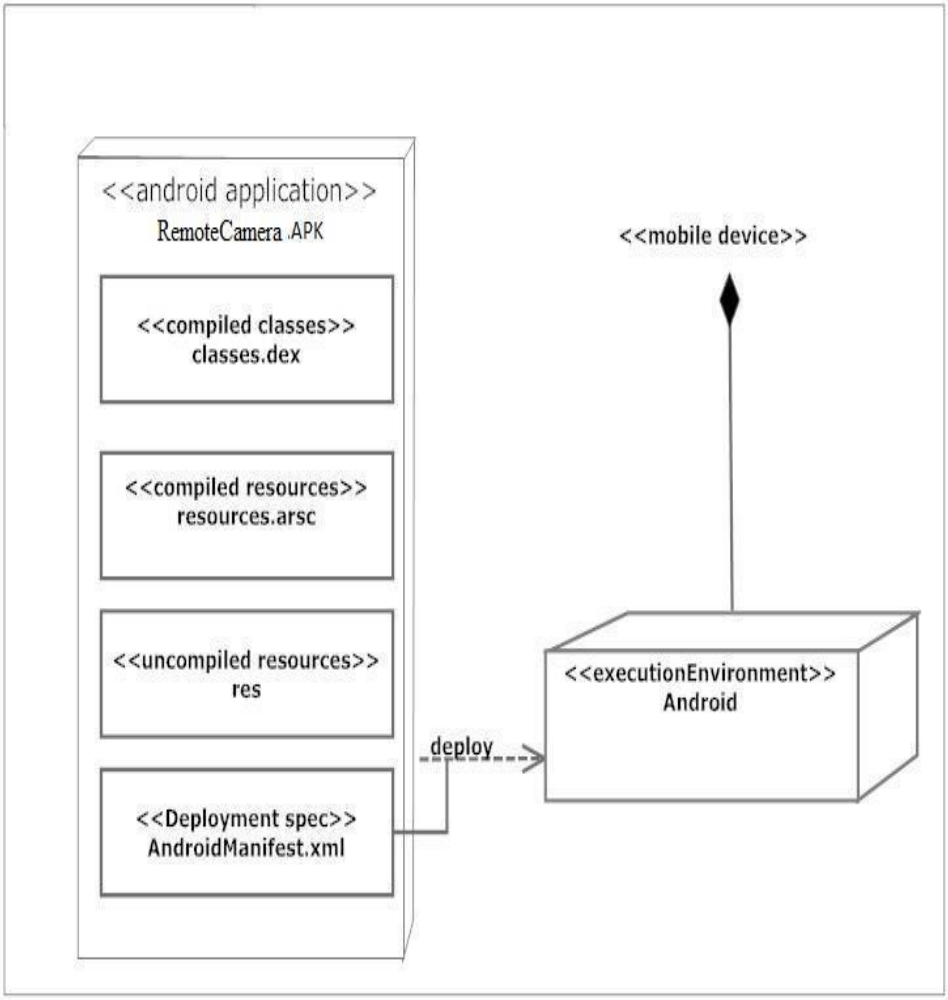


Fig 4.2.5: Deployment Diagram

4.2.6 Package Diagram

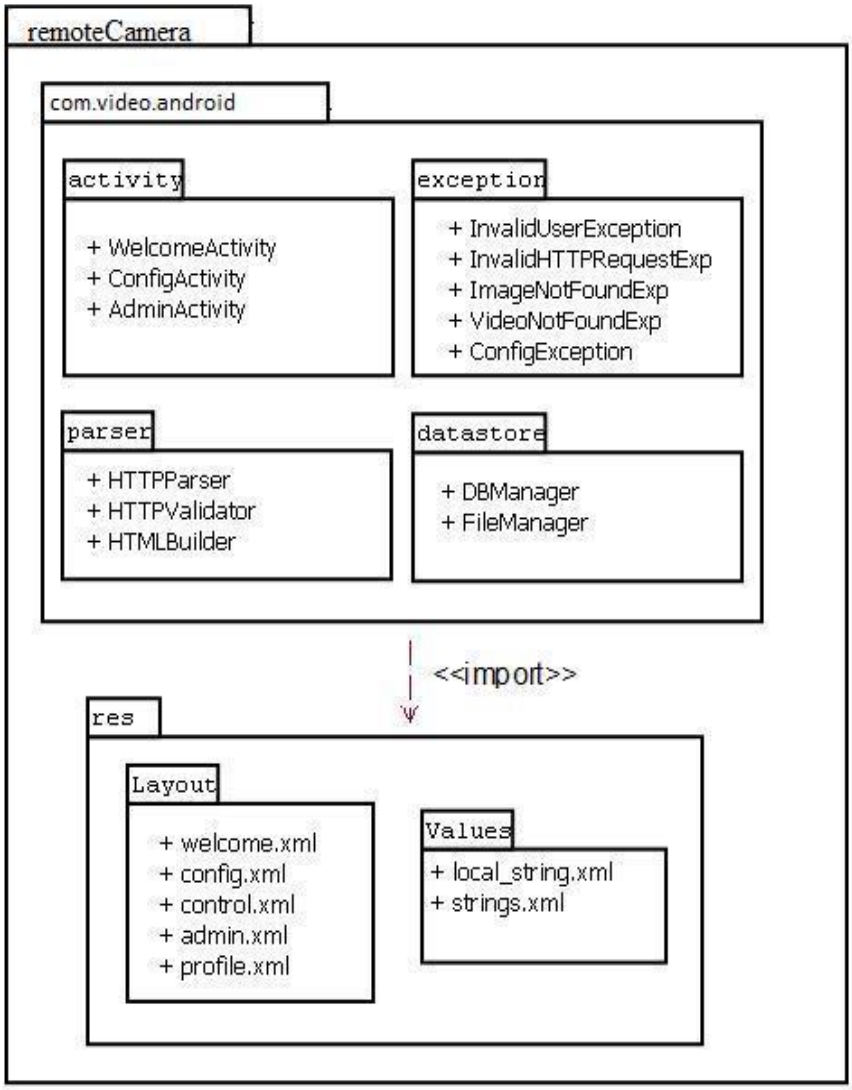


Fig 4.2.6 Package Diagram

Chapter 5

TECHNICAL SPECIFICATION

5.1 Technology Details Used In The Project

5.1.1 Core Java

Over the ages people have used tools to help them accomplish tasks, but lately their tools have been getting smarter and interconnected. Microprocessors have appeared inside many commonly used items, and increasingly, they have been connected to networks. As the heart of personal computers and workstations, for example, microprocessors have been routinely connected to networks. They have also appeared inside devices with more specific functionality than the personal computer or the workstation. Televisions, VCRs, audio components, fax machines, scanners, printers, cell phones, personal digital assistants, pagers, and wrist-watches--all have been enhanced with microprocessors; most have been connected to networks. Given the increasing capabilities and decreasing costs of information processing and data networking technologies, the network is rapidly extending its reach.

The emerging infrastructure of smart devices and computers interconnected by networks represents a new environment for software--an environment that presents new challenges and offers new opportunities to software developers. Java is well suited to help software developers meet challenges and seize opportunities presented by the emerging computing environment, because Java was designed for networks. Its suitability for networked environments is inherent in its architecture, which enables secure, robust, platform- independent programs to be delivered across networks and run on a great variety of computers and devices.

The Architecture

Java's architecture arises out of four distinct but interrelated technologies:

- the Java programming language
- the Java class file format
- the Java Application Programming Interface

When you write and run a Java program, you are tapping the power of these four technologies. You express the program in source files written in the Java programming language, compile the source to Java class files, and run the class files on a Java virtual machine. When you write your program, you access system resources (such as I/O, for example) by calling methods in the classes that implement the Java Application Programming Interface, or Java API. As your program runs, it fulfills your program's Java API calls by invoking methods in class files that implement the Java API. You can see the relationship between these four parts in Figure 1-1.

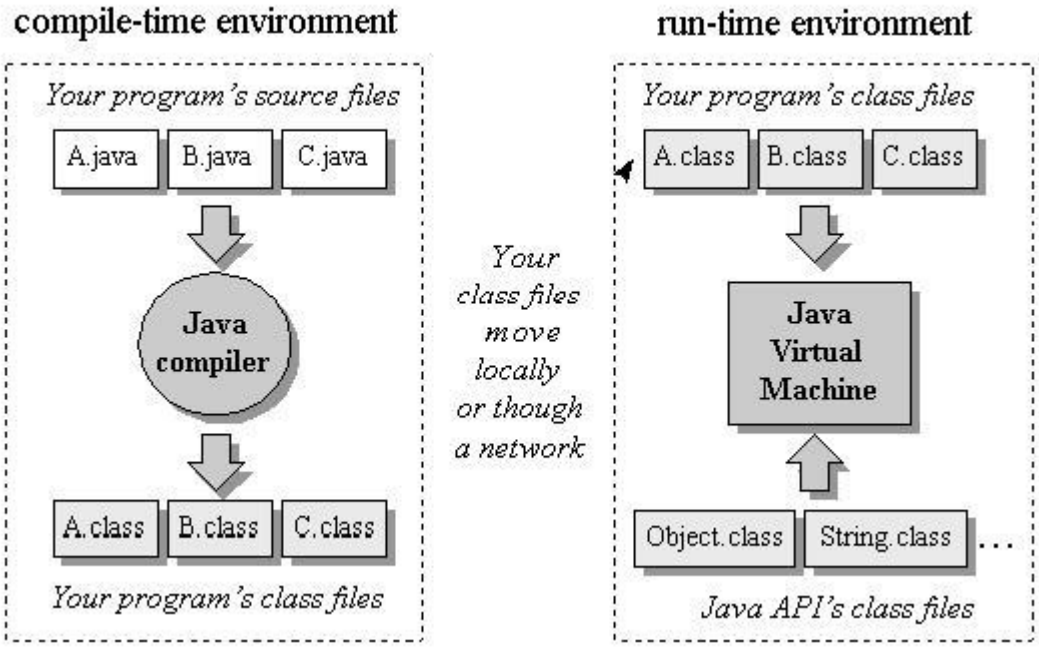


Fig.5-1. The Java programming environment

Together, the Java virtual machine and Java API form a "platform" for which all Java programs are compiled. In addition to being called the *Java runtime system*, the combination of the Java virtual machine and Java API is called the *Java Platform* (or, starting with version 1.2, the *Java 2 Platform*). Java programs can run on many different kinds of computers because the Java Platform can itself be implemented in software. As you can see in Figure 1-2, a Java program can run anywhere the Java Platform is present.

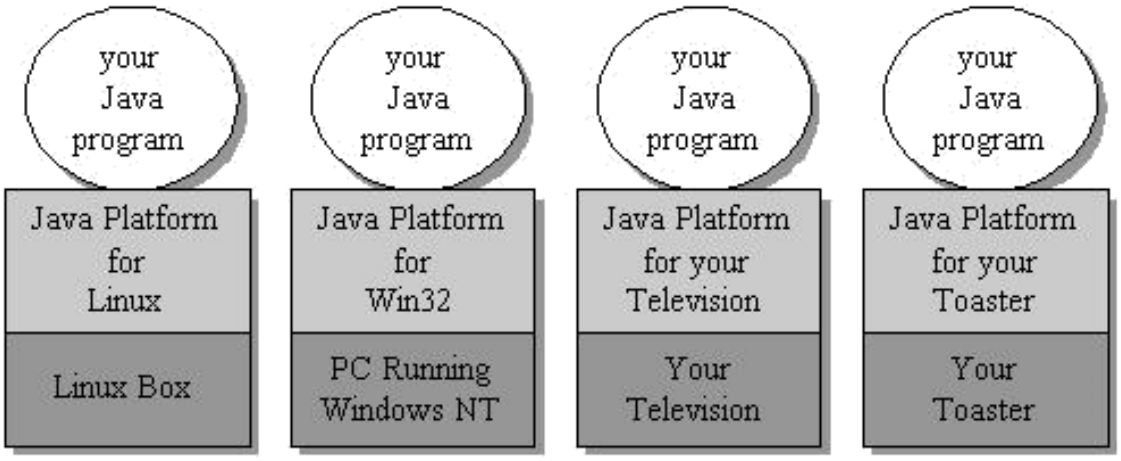


Fig.5-2. Java programs run on top of the Java Platform

The Java Virtual Machine

At the heart of Java's network-orientation is the Java virtual machine, which supports all three prongs of Java's network-oriented architecture: platform independence, security, and network-mobility.

The Java virtual machine is an abstract computer. Its specification defines certain features every Java virtual machine must have, but leaves many choices to the designers of each implementation. For example, although all Java virtual machines must be able to execute Java bytecodes, they may use any technique to execute them. Also, the specification is flexible enough to allow a Java virtual machine to be implemented either completely in software or to varying degrees in hardware. The flexible nature of the Java virtual machine's specification enables it to be implemented on a wide variety of computers and devices.

A Java virtual machine's main job is to load class files and execute the bytecodes they contain. As you can see in Figure 1-3, the Java virtual machine contains a *class loader*, which loads class files from both the program and the Java API. Only those class files from the Java API that are actually needed by a running program are loaded into the virtual machine. The bytecodes are executed in an *execution engine*.

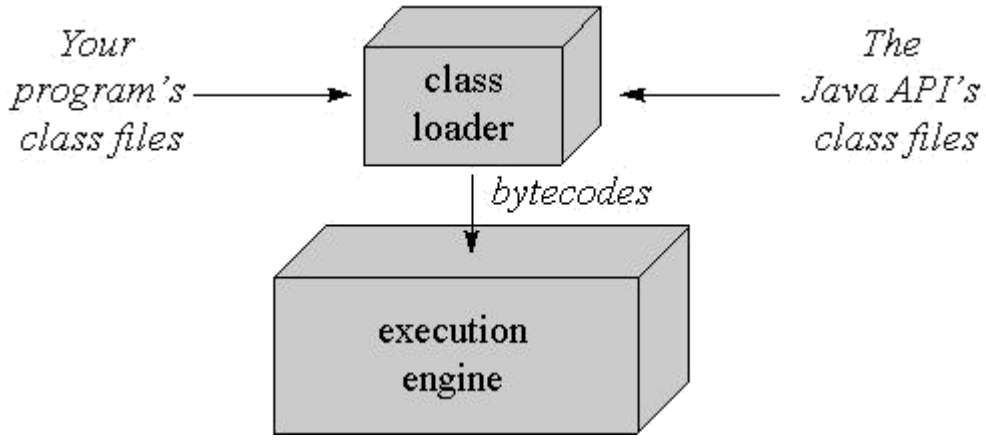


Fig. 5.3- A basic block diagram of the Java virtual machine

The execution engine is one part of the virtual machine that can vary in different implementations. On a Java virtual machine implemented in software, the simplest kind of execution engine just interprets the bytecodes one at a time. Another kind of execution engine, one that is faster but requires more memory, is a just-in-time compiler. In this scheme, the bytecodes of a method are compiled to native machine code the first time the method is invoked. The native machine code for the method is then cached, so it can be re-used the next time that same method is invoked. A third type of execution engine is an *adaptive optimizer*. In this approach, the virtual machine starts by interpreting bytecodes, but monitors the activity of the running program and identifies the most heavily used areas of code. As the program runs, the virtual machine compiles to native and optimizes just these heavily used areas. The rest of the of code, which is not heavily used, remain as bytecodes which the virtual machine continues to interpret. This adaptive optimization approach enables a Java virtual machine to spend typically 80 to 90% of its time executing highly optimized native code, while requiring it to compile and optimize only the 10 to 20% of the code that really matters to performance. Lastly, on a Java virtual machine built on top of a chip that executes Java byte codes natively, the execution engine is actually embedded in the chip.

Sometimes the Java virtual machine is called the Java interpreter; however, given the various ways in which byte codes can be executed, this term can be misleading.

While "Java interpreter" is a reasonable name for a Java virtual machine that interprets byte codes, virtual machines also use other techniques (such as just-in-time compiling) to execute byte codes. Therefore, although all Java interpreters are Java virtual machines, not all Java virtual machines are Java interpreters.

When running on a Java virtual machine that is implemented in software on top of a host operating system, a Java program interacts with the host by invoking *native methods*. In Java, there are two kinds of methods: Java and native. A Java method is written in the Java language, compiled to byte codes, and stored in class files. A native method is written in some other language, such as C, C++, or assembly, and compiled to the native machine code of a particular processor. Native methods are stored in a dynamically linked library whose exact form is platform specific. While Java methods are platform independent, native methods are not. When a running Java program calls a native method, the virtual machine loads the dynamic library that contains the native method and invokes it. As you can see in Figure 1-4, native methods are the connection between a Java program and an underlying host operating system.

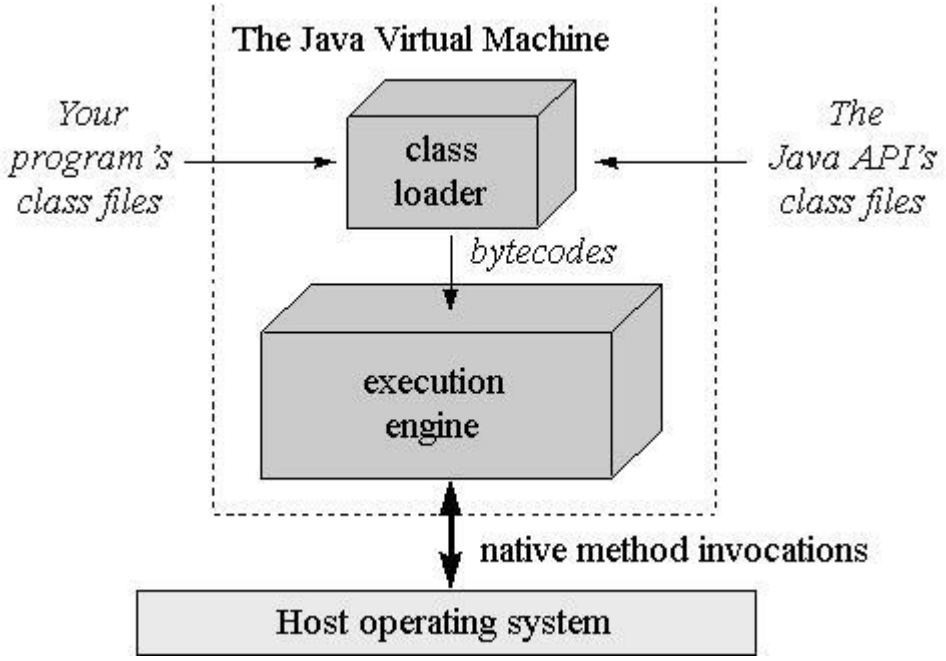


Fig. 5.4- A Java virtual machine implemented in software on top of a host operating system

You can use native methods to give your Java programs direct access to the resources of the underlying operating system. Their use, however, will render your program platform specific, because the dynamic libraries containing the native methods are platform specific. In addition, the use of native methods may render your program specific to a particular implementation of the Java Platform. One native method interface--the *Java Native Interface*, or *JNI*--enables native methods to work with any Java Platform implementation on a particular host computer. Vendors of the Java Platform, however, are not necessarily required to support JNI. They may provide their own proprietary native method interfaces in addition to (or depending on their contract, in place of) JNI.

Java gives you a choice. If you want to access resources of a particular host that are unavailable through the Java API, you can write a platform-specific Java program that calls native methods. If you want to keep your program platform independent, however, you must access the system resources of the underlying operating system only through the Java API.

The Class Loader Architecture

One aspect of the Java virtual machine that plays an important role in both security and network-mobility is the class loader architecture. In the block diagrams of Figures 1-3 and 1-4, a single mysterious cube identifies itself as "the class loader," but in reality there may be more than one class loader inside a Java virtual machine. Thus the class loader cube of the block diagram actually represents a subsystem that may involve many class loaders. The Java virtual machine has a flexible class loader architecture that allows a Java application to load classes in custom ways.

A Java application can use two types of class loaders: a "bootstrap" class loader and user-defined class loaders. The bootstrap class loader (there is only one of them) is a part of the Java virtual machine implementation. For example, if a Java virtual machine is implemented as a C program on top of an existing operating system, then the bootstrap class loader will be part of that C program. The bootstrap class loader loads classes, including the classes of the Java API, in some default way, usually from the local disk. (The bootstrap class loader has also been called the primordial class loader, system class loader, or default class loader.

In 1.2, the name "system class loader" was given a new meaning, which is described in Chapter 3.)

At run-time, a Java application can install user-defined class loaders that load classes in custom ways, such as by downloading class files across a network. While the bootstrap class loader is an intrinsic part of the virtual machine implementation, user-defined class loaders are not. Instead, user-defined class loaders are written in Java, compiled to class files, loaded into the virtual machine, and instantiated just like any other object. They are really just another part of the executable code of a running Java application. You can see a graphical depiction of this architecture in Figure 1-5.

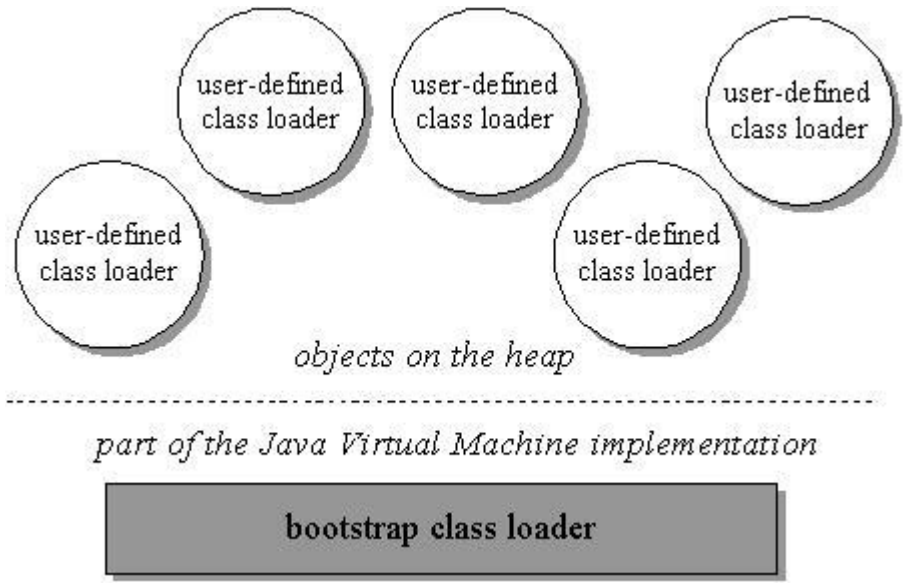


Fig. 5.5- Java's class loader architecture

Because of user-defined class loaders, you don't have to know at compile-time all the classes that may ultimately take part in a running Java application. User-defined class loaders enable you to dynamically extend a Java application at run-time. As it runs, your application can determine what extra classes it needs and load them through one or more user-defined class loaders. Because you write the class loader in Java, you can load classes in any manner expressible in Java code.

You can download them across a network, get them out of some kind of database, or even calculate them on the fly.For each class it loads, the Java virtual machine keeps track of which class loader--whether bootstrap or user-defined--loaded the class. When a loaded class first refers to another class, the virtual machine requests the referenc*ed* class from the same class loader that originally loaded the referenc*ing* class. For example, if the virtual machine loads class Volcano through a particular class loader, it will attempt to load any classes Volcano refers to through the same class loader. If Volcano refers to a class named Lava, perhaps by invoking a method in class Lava, the virtual machine will request Lava from the class loader that loaded Volcano. The Lava class returned by the class loader is dynamically linked with class Volcano.

Because the Java virtual machine takes this approach to loading classes, classes can by default only see other classes that were loaded by the same class loader. In this way, Java's architecture enables you to create multiple *name-spaces* inside a single Java application. Each class loader in your running Java program has its own name-space, which is populated by the names of all the classes it has loaded.

A Java application can instantiate multiple user-defined class loaders either from the same class or from multiple classes. It can, therefore, create as many (and as many different kinds of) user-defined class loaders as it needs. Classes loaded by different class loaders are in different name-spaces and cannot gain access to each other unless the application explicitly allows it. When you write a Java application, you can segregate classes loaded from different sources into different name-spaces. In this way, you can use Java's class loader architecture to control any interaction between code loaded from different sources. In particular, you can prevent hostile code from gaining access to and subverting friendly code.

One example of dynamic extension is the web browser, which uses user-defined class loaders to download the class files for applets across a network. A web browser fires off a Java application that installs a user-defined class loader--usually called an *applet class loader*-- that knows how to request class files from an HTTP server. Applets are an example of dynamic extension, because the Java application doesn't know when it starts which class files the browser will ask it to download across the network.

The class files to download are determined at run-time, as the browser encounters pages that contain Java applets. The Java application started by the web browser usually creates a different user-defined class loader for each location on the network from which it retrieves class files. As a result, class files from different sources are loaded by different user-defined class loaders. This places them into different name-spaces inside the host Java application. Because the class files for applets from different sources are placed in separate name- spaces, the code of a malicious applet is restricted from interfering directly with class files downloaded from any other source.

By allowing you to instantiate user-defined class loaders that know how to download class files across a network, Java's class loader architecture supports network-mobility. It supports security by allowing you to load class files from different sources through different user-defined class loaders. This puts the class files from different sources into different name-spaces, which allows you to restrict or prevent access between code loaded from different sources.

The Java Class File

The Java class file helps make Java suitable for networks mainly in the areas of platform-independence and network-mobility. Its role in platform independence is serving as a binary form for Java programs that is expected by the Java virtual machine but independent of underlying host platforms. This approach breaks with the tradition followed by languages such as C or C++. Programs written in these languages are most often compiled and linked into a single binary executable file specific to a particular hardware platform and operating system. In general, a binary executable file for one platform won't work on another. The Java class file, by contrast, is a binary file that can be run on any hardware platform and operating system that hosts the Java virtual machine.

When you compile and link a C++ program, the executable binary file you get is specific to a particular target hardware platform and operating system because it contains machine language specific to the target processor. A Java compiler, by contrast, translates the instructions of the Java source files into byte codes, the "machine language" of the Java virtual machine. In addition to processor-specific machine language, another platform-dependent attribute of a traditional binary executable file is the byte order of integers.

In executable binary files for the Intel X86 family of processors, for example, the byte order is *little-endian*, or lower order byte first. In executable files for the PowerPC chip, however, the byte order is *big-endian*, or higher order byte first. In a Java class file, byte order is big-endian irrespective of what platform generated the file and independent of whatever platforms may eventually use it.

In addition to its support for platform independence, the Java class file plays a critical role in Java's architectural support for network-mobility. First, class files were designed to be compact, so they can more quickly move across a network. Also, because Java programs are dynamically linked and dynamically extensible, class files can be downloaded as needed. This feature helps a Java application manage the time it takes to download class files across a network, so the end-user's wait time can be kept to a minimum.

The Java API

The Java API helps make Java suitable for networks through its support for platform independence and security. The Java API is set of runtime libraries that give you a standard way to access the system resources of a host computer. When you write a Java program, you assume the class files of the Java API will be available at any Java virtual machine that may ever have the privilege of running your program. This is a relatively safe assumption because the Java virtual machine and the class files for the Java API are the required components of any implementation of the Java Platform. When you run a Java program, the virtual machine loads the Java API class files that are referred to by your program's class files. The combination of all loaded class files (from your program and from the Java API) and any loaded dynamic libraries (containing native methods) constitute the full program executed by the Java virtual machine.

The class files of the Java API are inherently specific to the host platform. The API's functionality must be implemented expressly for a particular platform before that platform can host Java programs. To access the native resources of the host, the Java API calls native methods. As you can see in Figure 1-6, the class files of the Java API invoke native methods so your Java program doesn't have to. In this manner, the Java API's class files provide a Java program with a standard, platform-independent interface to the underlying host.

To the Java program, the Java API looks the same and behaves predictably no matter what platform happens to be underneath. Precisely because the Java virtual machine and Java API are implemented specifically for each particular host platform, Java programs themselves can be platform independent.

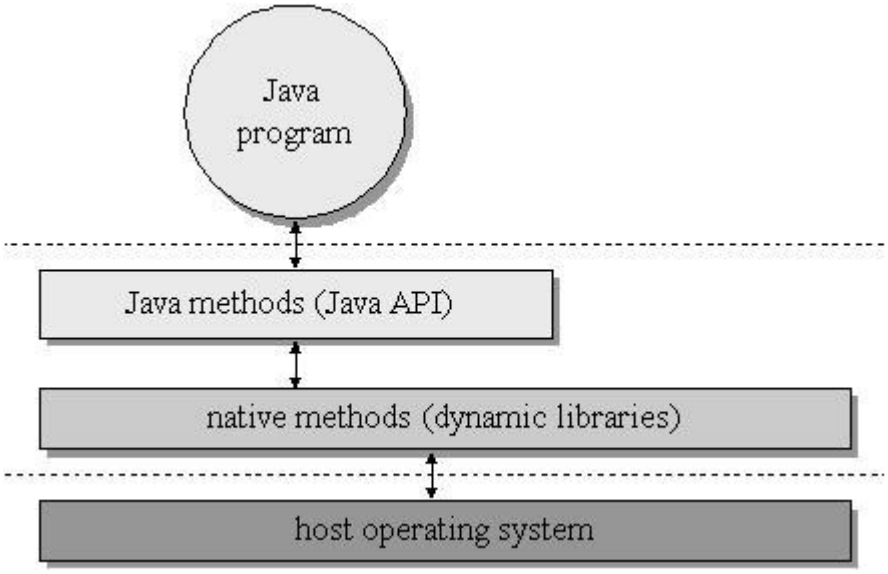


Fig. 5.6- A platform-independent Java program

The internal design of the Java API is also geared towards platform independence. For example, the graphical user interface libraries of the Java API, the Abstract Windows Toolkit (or AWT) and Swing, are designed to facilitate the creation of user interfaces that work on all platforms. Creating platform- independent user interfaces is inherently difficult, given that the native look and feel of user interfaces vary greatly from one platform to another. The AWT library's architecture does not coerce implementations of the Java API to give Java programs a user interface that looks exactly the same everywhere. Instead, it encourages implementations to adopt the look and feel of the underlying platform. The Swing library offers even more flexibility -- enabling the look and feel to be chosen by the programmer. Also, because the size of fonts, buttons, and other user interface components will vary from platform to platform, the AWT and Swing include *layout managers* to position the elements of a window or dialog box at run- time.

Rather than forcing you to indicate exact X and Y coordinate for the various elements that constitute, say, a dialog box, the layout manager positions them when your dialog box is displayed. With the aim of making the dialog look its best on each platform, the layout manager will very likely position the dialog box elements slightly differently on different platforms. In these ways and many others, the internal architecture of the Java API is aimed at facilitating the platform independence of the Java programs that use it.

In addition to facilitating platform independence, the Java API contributes to Java's security model. The methods of the Java API, before they perform any action that could potentially be harmful (such as writing to the local disk), check for permission. In Java releases prior to 1.2, the methods of the Java API checked permission by querying the *security manager*. The security manager is a special object that defines a custom security policy for the application. A security manager could, for example, forbid access to the local disk. If the application requested a local disk write by invoking a method from the pre-1.2 Java API, that method would first check with the security manager. Upon learning from the security manager that disk access is forbidden, the Java API would refuse to perform the write. In Java 1.2, the job of the security manager was taken over by the *access controller*, a class that performs stack inspection to determine whether the operation should be allowed. (For backwards compatibility, the security manager still exists in Java 1.2.) By enforcing the security policy established by the security manager and access controller, the Java API helps to establish a safe environment in which you can run potentially unsafe code.

5.1.2 XML

- XML stands for Extensible Markup Language
- XML is markup language much like HTML
- XML was designed to carry data ,not to display data
- XML tags are not predefined. You must defined your own tags
- XML is designed to be self-descriptive
- XML is W3C Recommendation

5.1.3 Android

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

Features:

1. Application framework enabling reuse and replacement of components
2. Dalvik virtual machine optimized for mobile devices
3. Integrated browser based on the open source WebKit engine
4. Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
5. SQLite for structured data storage
6. Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
7. GSM Telephony (hardware dependent)
8. Bluetooth, EDGE, 3G, and Wi-Fi (hardware dependent)
9. Camera, GPS, compass, and accelerometer (hardware dependent)

Chapter 6

PROJECT ESTIMATE, SCHEDULE & TEAM STRUCTURE

6.1 Process based Estimates

6.1.1 Cost Estimation:

It is often called “cost/ benefit” analysis, it evaluates saving & financial benefit from new system ,as well as evaluate the cost of system on the development & installation ,operation & maintenance .

Capital Budget: It is based on hardware cost,s/w cost ,design analysis cost & cost of client time. As far as concern with project hardware cost just required a virtual device which is easily available & free as well &software cost is for java,JMF which are free available .

6.2 Team Structure:

People:

Total no of people: 4

1. Khot Harish Shridhar
2. Khatal Sonali Bhanudas
3. Gote Swati Ramdas
4. Pandarge Sangmesh Shivkant

6.3ProjectSchedule:

6.3.1 Network Diagram

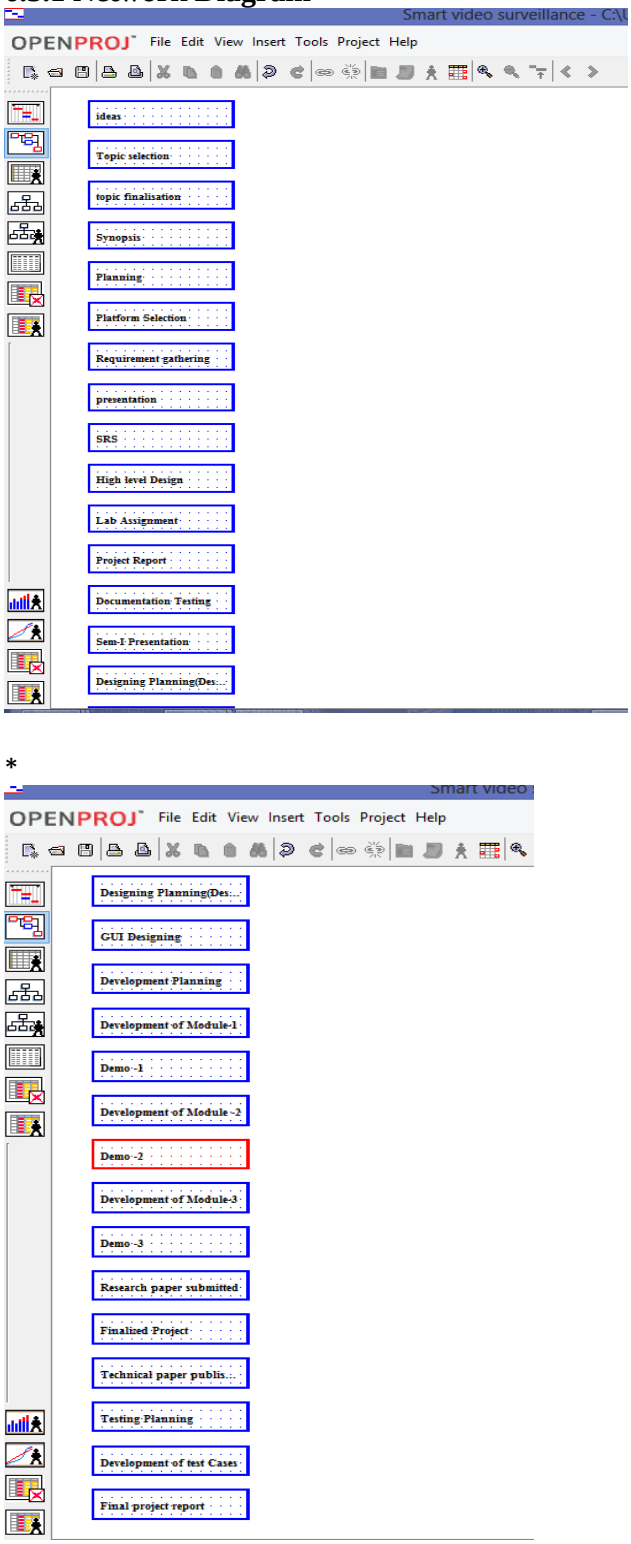
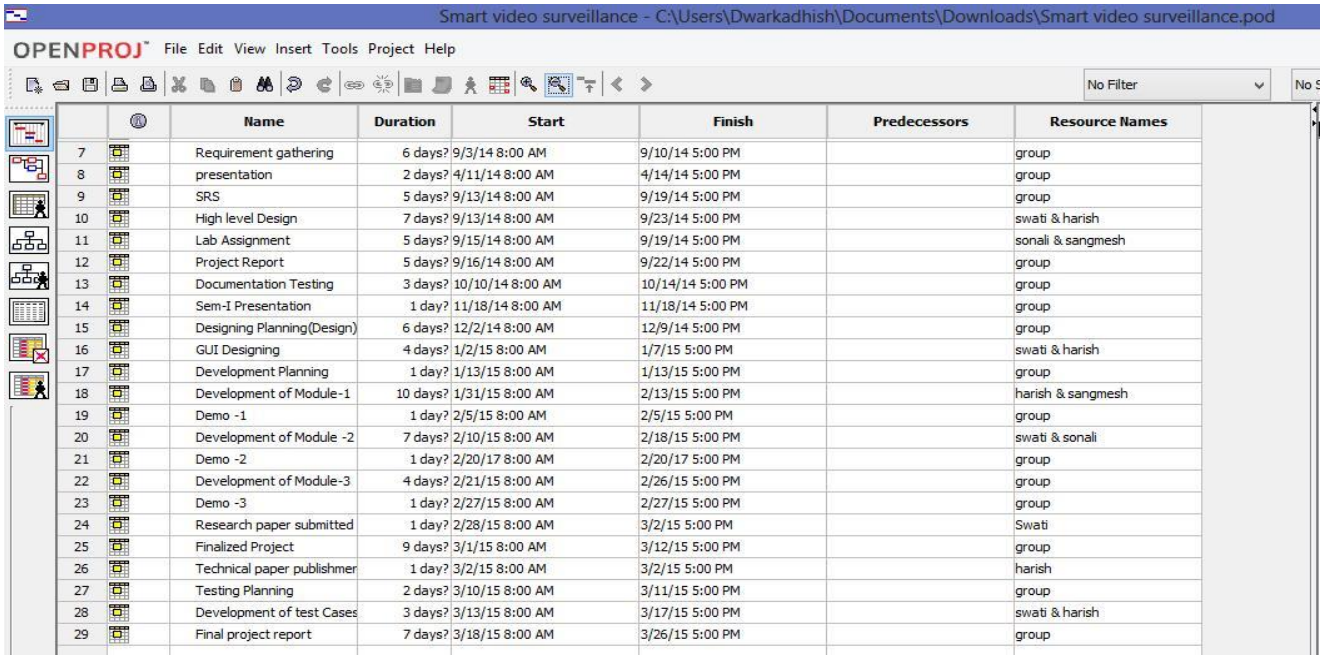


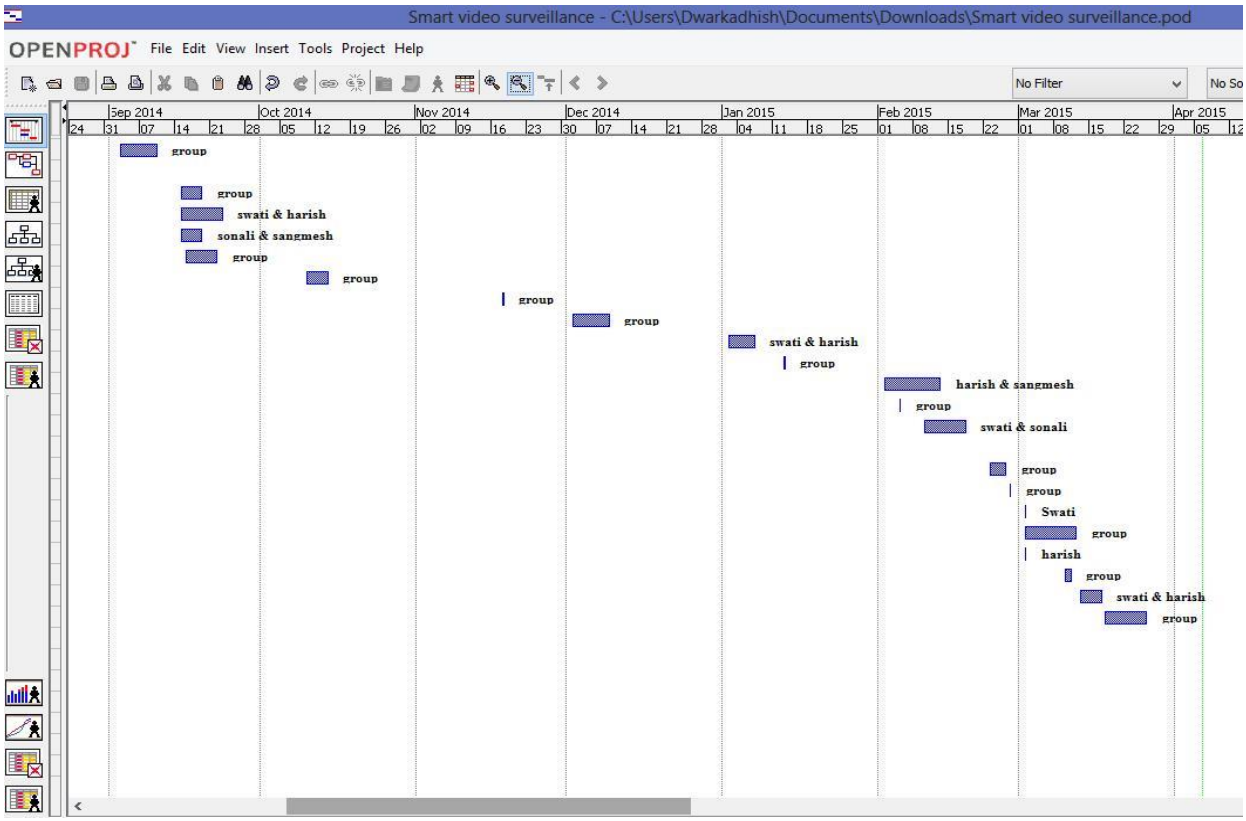
Fig.6.3.1-Network diagram

6.3.2 Project Planning



	Name	Duration	Start	Finish	Predecessors	Resource Names
7	Requirement gathering	6 days?	9/3/14 8:00 AM	9/10/14 5:00 PM		group
8	presentation	2 days?	4/11/14 8:00 AM	4/14/14 5:00 PM		group
9	SRS	5 days?	9/13/14 8:00 AM	9/19/14 5:00 PM		group
10	High level Design	7 days?	9/13/14 8:00 AM	9/23/14 5:00 PM		swati & harish
11	Lab Assignment	5 days?	9/15/14 8:00 AM	9/19/14 5:00 PM		sonali & sangmesh
12	Project Report	5 days?	9/16/14 8:00 AM	9/22/14 5:00 PM		group
13	Documentation Testing	3 days?	10/10/14 8:00 AM	10/14/14 5:00 PM		group
14	Sem-I Presentation	1 day?	11/18/14 8:00 AM	11/18/14 5:00 PM		group
15	Designing Planning(Design)	6 days?	12/2/14 8:00 AM	12/9/14 5:00 PM		group
16	GUI Designing	4 days?	1/2/15 8:00 AM	1/7/15 5:00 PM		swati & harish
17	Development Planning	1 day?	1/13/15 8:00 AM	1/13/15 5:00 PM		group
18	Development of Module-1	10 days?	1/31/15 8:00 AM	2/13/15 5:00 PM		harish & sangmesh
19	Demo -1	1 day?	2/5/15 8:00 AM	2/5/15 5:00 PM		group
20	Development of Module -2	7 days?	2/10/15 8:00 AM	2/18/15 5:00 PM		swati & sonali
21	Demo -2	1 day?	2/20/17 8:00 AM	2/20/17 5:00 PM		group
22	Development of Module-3	4 days?	2/21/15 8:00 AM	2/26/15 5:00 PM		group
23	Demo -3	1 day?	2/27/15 8:00 AM	2/27/15 5:00 PM		group
24	Research paper submitted	1 day?	2/28/15 8:00 AM	3/2/15 5:00 PM		Swati
25	Finalized Project	9 days?	3/1/15 8:00 AM	3/12/15 5:00 PM		group
26	Technical paper publishmer	1 day?	3/2/15 8:00 AM	3/2/15 5:00 PM		harish
27	Testing Planning	2 days?	3/10/15 8:00 AM	3/11/15 5:00 PM		group
28	Development of test Cases	3 days?	3/13/15 8:00 AM	3/17/15 5:00 PM		swati & harish
29	Final project report	7 days?	3/18/15 8:00 AM	3/26/15 5:00 PM		group

6.3.3Gantt Chart



Chapter 7

SOFTWARE IMPLEMENTATION

7.1 Introduction

7.1.1 JDK

J2SE (Java 2 Standard Edition) Java would be the required as language for development of the project. JDK is the development kit used to compile java programs. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

7.1.2 Eclipse

In computer programming eclipse is a Multilanguage s/w development environment comprising a base workspace & Extensible plug-in system for customizing the environment it is written mostly in java.

7.2 DATABASE

DATABASE Details:

SQLite Database:

SQLite is an ACID-compliant embedded relational database management system contained in a relatively small (~225 kB) C programming library. The source code for SQLite is in the public domain.

Unlike client-server database management systems, the SQLite engine is not a standalone process with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple function calls, which reduces latency in database access as function calls within a single process are more efficient than inter-process communication.

The entire database (definitions, tables, indices, and the data itself) is stored as a single cross-platform file on a host machine.

This simple design is achieved by locking the entire database file during writing.

DB Schema

This application contains the two tables. Namely user table and config table.

User table is having seven fields namely userId, username, userAddress, userPhoneNumber, userEmailId, username and password.

When user follow the registration process on browser he has to fill all these details. Then using the url pattern all the fields information will be stored into local database ie sqlite database.

User id field accepts the integer values, its type is declared as integer in database. The username, password, userAddress and all the other fields accepts the string values. Because their type is described as string in database.

Database Name: httpServerDB

UserTable:

Name	Type	Null
userId	Int	No
Username	Varchar(20)	No
userAddress	Varchar(20)	No
userPhoneNumber	Varchar(20)	No
userEmailId	Varchar(20)	No
user_name	Varchar(20)	No
Password	Varchar(20)	No

Following classes is responsible to deal with configuration table:

- UserDBHelper class:
    - This class is the helper class and responsible for application to get and set User parameters.

- UserDBAdapter class:
    - This class actually fire database query and fetch data from our database.

```
public int insertData(UserBean userProfileBean)
{
    ContentValues cValues = new ContentValues();
    cValues.put(COLUMN_USER_NAME, userProfileBean.getUserName());
    cValues.put(COLUMN_USER_ADDRESS, userProfileBean.getUserAddress());
    cValues.put(COLUMN_USER_PHONE_NUMBER, userProfileBean.getUserPhoneNumber());
    cValues.put(COLUMN_USER_EMAIL_ID, userProfileBean.getUserEmailId());
    cValues.put(COLUMN_USERNAME, userProfileBean.getUsername());
    cValues.put(COLUMN_PASSWORD, userProfileBean.getPassword());

    return (int)mySqliteDB.insert(USER_TABLE_NAME, null, cValues);
}
```

The second table is config table which is having three fields namely property, value, property type. All three fields accept string values only.

Configuration table:

Name	Type	Null
Property	Varchar(20)	No
Value	Varchar(20)	No
Propertytype	Varchar(20)	No

Following classes is responsible to deal with configuration table:

- ConfigDBHelper class:
    - This class is the helper class and responsible for application to get and set configuration parameters.

- ConfigBean class:

    - This class is the place holder class and contains only getter setter methods to access values. It Bean class contains the structure of the respective table. Structure includes the all the fields of the table with their types.(string, int, double etc). Bean class contains all the setter and getter methods. Like setUserName(), getUserName(), setId(), getId() etc.

```
public class ConfigBean {
    private String property;
    private String value;
    private String propertyType;

    public String getPropertyType() {
        return propertyType;
    }
    public void setPropertyType(String propertyType) {
        this.propertyType = propertyType;
    }
```

- ConfigDBAdapter class:

    - This class actually fire database query and fetch data from our database. It contains all the queries for the particular operation like insert(), update(), delete() etc. From these three classes the adapter class actually communicates with the database. This class fires the queries on the database.

We create a subclass implementing onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and optionally on Open(SQLiteDatabase), and this class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary. Transactions are used to make sure the database is always in a sensible state.

This class makes it easy for ContentProvider implementations to defer opening and upgrading the database until first use, to avoid blocking application startup with long-running database upgrades.

1. public long insert (String table, String nullColumnHack, ContentValues values)

Added in API level 1

Convenience method for inserting a row into the database.

Parameters

Table	the table to insert the row into
nullColumnHack	optional; may be null. SQL doesn't allow inserting a completely empty row without naming at least one column name. If your provided values is empty, no column names are known and an empty row can't be inserted. If not set to null, the nullColumnHack parameter provides the name of nullable column name to explicitly insert a NULL into in the case where your values is empty.
Values	this map contains the initial column values for the row. The keys should be the column names and the values the column values

Returns

- the row ID of the newly inserted row, or -1 if an error occurred

2. public
   int update (String table, ContentValues values, String whereClause, String[] whereArgs)

Convenience method for updating rows in the database.

Parameters

Table	the table to update in
Values	a map from column names to new column values. null is a valid value that will be translated to NULL.

whereClause the optional WHERE clause to apply when updating. Passing null will update all rows.

whereArgs You may include ?s in the where clause, which will be replaced by the values from whereArgs. The values will be bound as Strings.

Returns

- the number of rows affected

3. public abstract int getColumnIndex(String columnName)

Returns the zero-based index for the given column name, or -1 if the column doesn't exist. If you expect the column to exist usegetColumnIndexOrThrow(String) instead, which will make the error more clear.

Parameters

columnName:- the name of the target column.

Returns

- the zero-based column index for the given column name, or -1 if the column name does not exist.

4. Public abstract Boolean isAfterLast()

Returns whether the cursor is pointing to the position after the last row.

Returns

- whether the cursor is after the last result.

Public abstract Boolean moveToFirst()

Move the cursor to the first row.

This method will return false if the cursor is empty.

Returns

- Whether the move succeeded.

7.3 Classes Description:

Our project contains following classes:

1. MainActivity class
2. CameraClass class
3. ConfigActivity class
4. MyHTTP class
5. HelloServer class
6. CameraPreview class
7. ConfigBean class
8. UserBean class
9. BaseAdapter class

Classes description is given bellow:

- MainActivity class:
  - MainActivity class is the launcher activity class. Using setContentView(R.layout.activity_main) method we are setting activity_main xml layout to this activity.

```
<activity
    android:name="com.android.mk.httpserver.main.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

    - In this activity there are three buttons on the activity_main layout file and their names are Start Server button, Stop Server button and Start Streaming button.
    - Inside onClick() method of the Start Server button we are calling the HelloServer class.

- HelloServer class:
    - HelloServer class is the socket listener class listen on the port 8088.

```
public HelloServer(MainActivity mContext) {
    super(8088);
    this.mContext = mContext;

}
```

    - In HelloServer class is we write a switch case for particular pages

- MyHTTP class:
    - MyHTTP class constructs an HTTP server on given hostname and port.

- CameraClass class:
    - When user click the Start Streaming button CameraClass class starts and opens the camera in the video mode.

- CameraPreview class:
    - CameraPreview class is the callback class.

- ContactOperation:
    - This class is call when we click on insert contact. In this class we write method for insert contact.

```
public class ContactOperation {

    private final static String TAG = "com.android.insertcetetecontavt";

    @SuppressWarnings("unused")
    public static void Insert2Contacts(Context ctx, String nameSurname,
            String telephone) {
        if (!isTheNumberExistsinContacts(ctx, telephone)) {
            ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
            int rawContactInsertIndex = ops.size();

            ops.add(ContentProviderOperation.newInsert(RawContacts.CONTENT_URI)
                    .withValue(RawContacts.ACCOUNT_TYPE, null)
                    .withValue(RawContacts.ACCOUNT_NAME, null).build());
            ops.add(ContentProviderOperation
                    .newInsert(ContactsContract.Data.CONTENT_URI)
                    .withValueBackReference(
                            ContactsContract.Data.RAW_CONTACT_ID,
                            rawContactInsertIndex)
```

UserBean class

- In this class we provide just getters and setters. This class is used just as place holder.

```
private int userId;
private String userName;
private String userAddress;
private String userPhoneNumber;
private String userEmailId;
private String username;
private String password;

public int getUserId() {
    return userId;
}
public void setUserId(int userId) {
    this.userId = userId;
```

BaseAdapter class

- This is generic class used to create database and tables. For this we have to extend our class with SqliteOpenHelper.

```
public class BaseDBAdapter extends SQLiteOpenHelper{
private static String DATABASE_NAME = "httpServerDB";

    public static String USER_TABLE_NAME = "userTable";
    public static String COLUMN_USER_ID = "userId";
    public static String COLUMN_USER_NAME = "userName";
    public static String COLUMN_USER_ADDRESS = "userAddress";
    public static String COLUMN_USER_PHONE_NUMBER = "userPhoneNumber";
    public static String COLUMN_USER_EMAIL_ID = "userEmailId";
    public static String COLUMN_USERNAME = "user_name";
    public static String COLUMN_PASSWORD = "password";
```

```
private String CREATE_USER_TABLE_QUERY = "create table " + USER_TABLE_NAME + " ( "+COLUMN_USER_ID+" integer primary key not null," +
    "+COLUMN_USER_NAME+" text , "+COLUMN_USER_ADDRESS+" text , "+COLUMN_USER_PHONE_NUMBER+" text, "+COLUMN_USER_EMAIL_ID+" text, " +
        ""+COLUMN_USERNAME+" text, "+COLUMN_PASSWORD+" text) ";
```

CHAPTER 8

SOFTWARE TESTING

8.1 Software Testing:

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

8.1.1 Verification:

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

8.1.2 Validation:

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

8.2 Basics of software testing:

There are two basics of software testing: blackbox testing and whitebox testing.

8.2.1 Blackbox Testing:

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

8.2.2 Whitebox Testing :

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

8.3 Types of testing:

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing:

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing:

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing:

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing:

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing:

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing:

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing:

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing:

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing:

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

Beta Testing:

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

8.4 TEST CASES:

Test Case ID	1
Test Case Description	Apk should get installed properly.
Steps	1.Install the application. 2.Open the application.
Test Case Result	Application should be successful installed.
Action Result	Application successfully started.
Status	Pass

Test Case ID	2
Test Case Description	After clicking on the android icon, application should be started.
Steps	1.Install the application. 2. First click on the menu 3. Search the application 4.Run application as android application.

Test Case Result	Application should be successfully started.
Action Result	Application successfully started.
Status	Pass

Test Case ID	3
Test Case Description	Functionality of start button.
Steps	1.Run application as android application. 2.Click on start button.
Test Case Result	Server should get started.
Action Result	Server should get started.
Status	Pass

Test Case ID	4
Test Case Description	Functionality of stop server Button.
Steps	1.Click on stop button. 2.Server should get stopped after clicking on the stop button
Test Case Result	Server stopped.
Action Result	Server stopped.
Status	Pass

Test Case ID	5
Test Case Description	Functionality of video streaming.
Steps	1.Click on application. 2.Click on video streaming button.
Test Case Result	Start the camera
Action Result	Start the camera
Status	Pass

Test Case ID	6
Test Case Description	Website is opened when hit with proper ip address and port number on browser
Steps	1.Open browser 2Give proper Ip address and [ort number and project name in url
Test Case Result	After hit on url index page is opened
Action Result	As expected
Status	Pass

Test Case ID	7
Test Case Description	System should only provide authority to admin
Steps	1. Start the server. 2. Do login with admin.
Test Case Result	Application should allow login to admin only

Action Result	As expected
Status	Pass

Test Case ID	8
Test Case Description	Check functionality after admin login
Steps	1.Open the website 2.Login with admin 3.Click on menu option
Test Case Result	After login as admin on next page we get all option for add user, video streaming
Action Result	As expected
Status	Pass

Test Case ID	9
Test Case Description	Check functionality for option menu
Steps	1.Open the website 2.Login with admin 3.Click on menu option
Test Case Result	After click on icon we will show next function as we required
Action Result	As expected
Status	Pass

Test Case ID	10
Test Case Description	Check functionality for SMS
Steps	1.Open the website 2.Login with admin 3.Click on menu option
Test Case Result	After camera capture any movement then immediately SMS is sent to user
Action Result	After camera capture any movement then immediately SMS is sent to user
Status	Pass

Test Case ID	11
Test Case Description	Check functionality for message HTTP Server link
Steps	1.Open the website
Test Case Result	When user got message he click on url then he directly reach to required page
Action Result	As expected
Status	Pass

Test Case ID	12
Test Case Description	On browser admin can login first.

Steps	1.Connect mobile hotspot to laptop. 2.Generate Ip address. 3. Enter the ip address of mobile in url of browser. 4.Enter 8088 port and connect to server.
Test Case Result	Admin login page displayed.
Action Result	Admin login page displayed.
Status	Pass

Test Case ID	13
Test Case Description	Admin can create new users.
Steps	1.Connect mobile hotspot to laptop. 2.Generate Ip address. 3. Enter the ip address of mobile in url of browser. 4.Enter 8088 port and connect to server. 5.Login with admin username and password.
Test Case Result	Login successful. Now create new user.
Action Result	New user created.
Status	Pass

Test Case ID	14
Test Case Description	Login with new user id and password.
Steps	1.Connect mobile hotspot to laptop. 2.Generate Ip address. 3. Enter the ip address of mobile in url of browser. 4.Connect to mobile server. 5.login with new user.
Test Case Result	Login successful.
Action Result	Login successful.
Status	Pass

Test Case ID	15
Test Case Description	System should not put the phone On flight mode while running.
Steps	1.Run application as android application. 2.Click on start button.
Test Case Result	Application should run and phone should on.
Action Result	Running properly.
Status	Pass

Test Case ID	16
Test Case Description	System should not hangs the phone.
Steps	1.Run application as android application. 2.Click on start button.
Test Case Result	Application should run.
Action Result	Application running and phone is also working properly.
Status	Pass

Test Case ID	17
Test Case Description	System should not disturb call activity.
Steps	1.Run application as android application. 2.Click on start button. 3.User can receive incoming calls.
Test Case Result	Application should not block incoming calls.
Action Result	User can receive calls.
Status	Pass

Test Case ID	18
Test Case Description	Application should not run as background service.
Steps	1.Run application as android application. 2.Click on start button. 3.User can receive incoming calls.
Test Case Result	Application should not run as background service.
Action Result	Application should not run as background service.
Status	Pass

Test Case ID	19
Test Case Description	System should not disturb message activity.
Steps	1.Run application as android application. 2.Click on start button. 3.User can receive incoming calls.
Test Case Result	Application should not block message
Action Result	Application should not block message
Status	Pass

Test Case ID	20
Test Case Description	System should consumes minimum battery
Steps	1.Run application as android application. 2.Click on start button. 3.User can receive incoming calls.
Test Case Result	Application should consumes minimum battery
Action Result	should consumes minimum battery
Status	Pass

CHAPETR 9

RESULTS

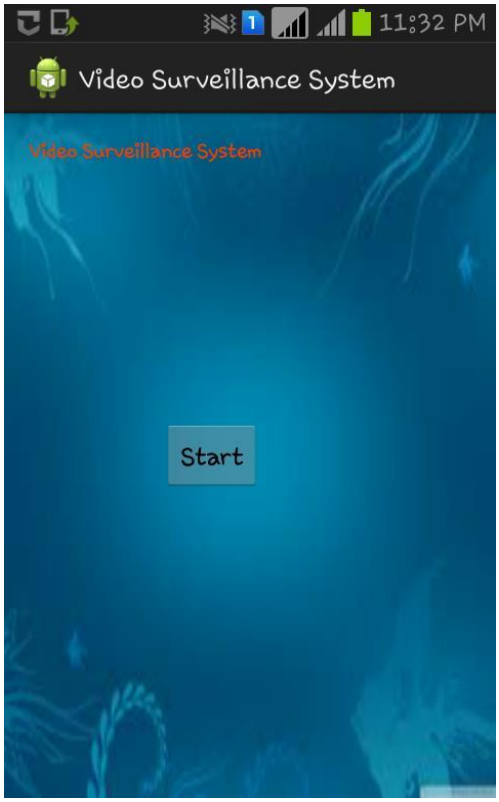


Fig 9.1: Start Window

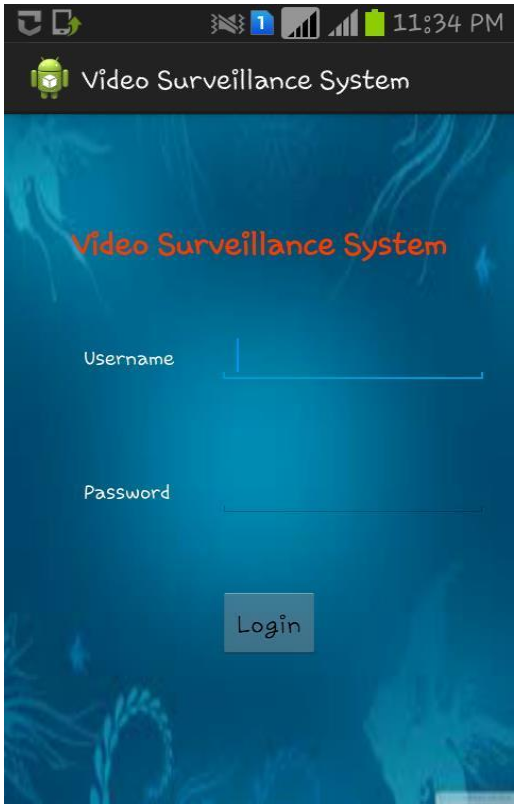


Fig.9.2: login window

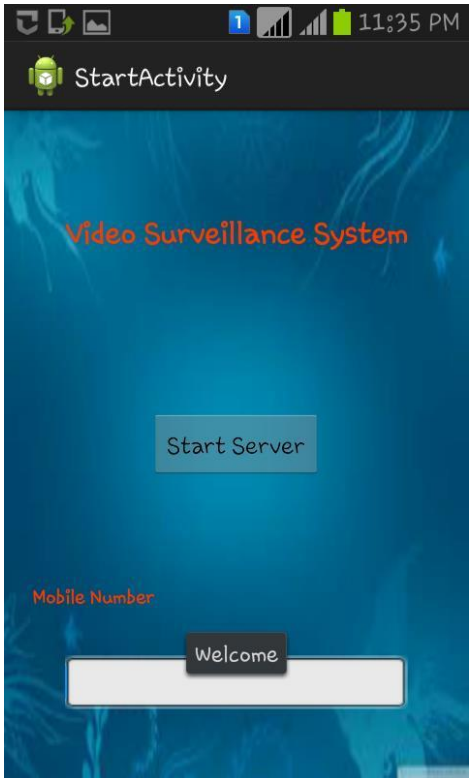


Fig9.3: start server window

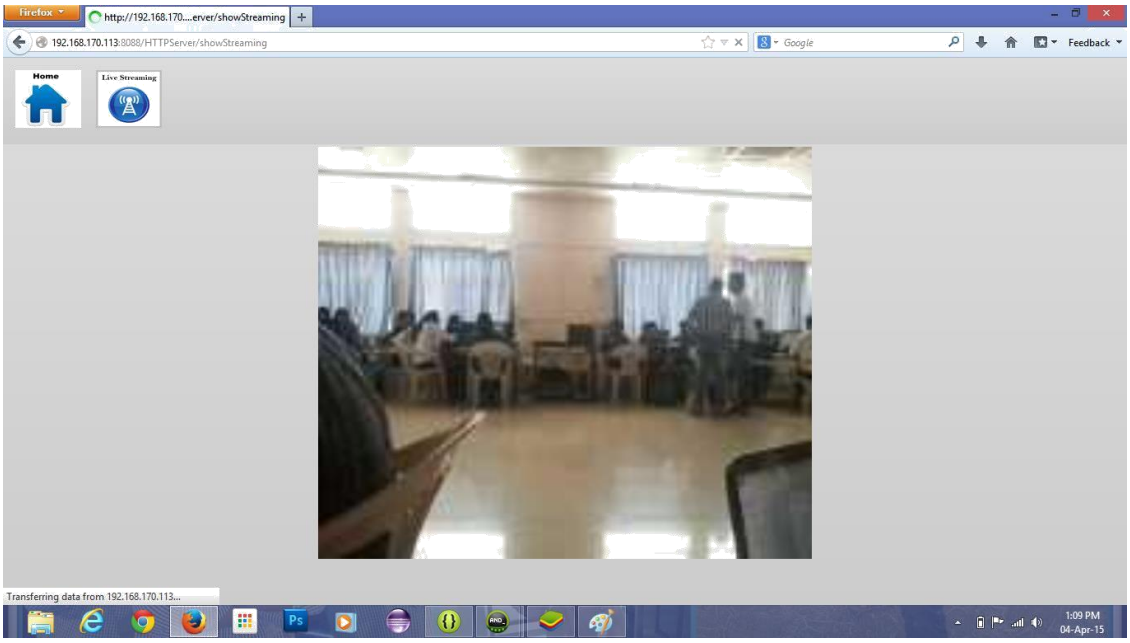


Fig 9.4: Web browser window

Chapter 10

DEPLOYMENT AND MAINTAINANCE

10.1 Installation And Uninstallation

10.1.1 Installation

1. Copy smart video surveillance.apk in device memory.

2. Just click on it and it will install on your device.

10.1.2 Uninstallation

1. Go to the user device setting

2.Go to the App Manager

3.Just click on application and uninstall it.

Chapter 11

CONCLUSION AND FUTURE SCOPE

Future scope:

In the future ,we are planning to upload our application to “Google play” so that everyone will be able to use our application. We are thinking of adding some new feature such as multi tabbing ,security, private browsing and the future ideas to it. We would like to work more and more on this project, to make it universally accepted.

Conclusion:

Smart Video Surveillance systems significantly contribute to situation awareness. Such systems transform video surveillance from data acquisition tool to information and intelligence acquisition systems. Real-time video analysis provides smart surveillance systems with the ability to react in real-time. Our system will senses the intrusion and sends notifications to authorized persons so that action can be taken in response to the intrusion.

REFERENCES

[1] D. Koller, K. Daniilidis, H. H. Nagel, Model-based object tracking in monocular sequences of road traffic scenes. International Journal of Computer Vision, Vol. 10, 1993, pp. 257-281.

[2] Yuri A. Ivanov and Aaron F. Bobick, Recognition of Multi-Agent Interaction in Video Surveillance, ICCV (1), pp. 169-176, 1999.

[3] Drew Ostheimer, Sebastien Lemay, Mohammed Ghazal, Dennis Mayisela, AishyAmer, Pierre F. Dagba: A Modular Distributed Video Surveillance System Over IP, 1-4244-0038-4 2006 IEEE CCECE/CCGEI, Ottawa, May 2006.

[4] Blanz and Vetter, Face recognition based on fitting 3D morphablemodel,IEEE PAMI, vol. 25, no. 9, pp. 1063-1074, Sept. 2003.

[5] R. Collins et al. A system for video surveillance and monitoring,VSAMFinaL Report, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-00-12, May 2000.Combat Zones That See, U.S. Government DARPA Project.M.W. Green, The appropriate and effective use of security technologies in U.S.schools, A guide for schools and law enforcement agencies, Sandia National

[6] Hampapur, S. Pankanti, A.W. Senior, Y-L. Tian, L. Brown, and R. Bolle, Facecataloger: Multi-scale imaging for relating identity to location, in Proc. IEEEConf. Advanced Video and Signal Based Surveillance, Miami, FL, 21-22 July 2003,pp. 13-20.

[7] Hampapur, L. Brown, J. Connell, M. Lu, H. Merkl, S. Pankanti, A. Senior,Shu, and Y. Tian, The IBM smart surveillance system, demonstration, Proc.IEEE, CVPR 2004.

[8] Hae-Min Moon, Sumg Bum Pan: A New Human Identification Method for Intelligent Video Surveillance System, 978-1-42447116-4/10/26.002010 IEEE