**PROJECT PHASE-I REPORT**

**ON**

Suspicious Activity Detection.

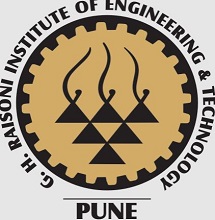
SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

**Under the Guidance of Prof. Sunita Vani**



**DEPARTMENT OF COMPUTER ENGINEERING**

**G H RAISONI COLLEGE OF ENGINEERING AND TECHNOLOGY**

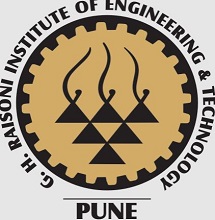
(An Institute affiliated to SPPU)

## WAGHOLI, PUNE-402207

**SAVITRIBAI PHULE PUNE UNIVERSITY 2021 -2022**

**G H Raisoni College of Engineering and Technology, Wagholi- Pune 402207**

(An Institute affiliated to SPPU)



# CERTIFICATE

This is to certify that the project report entitles

## “Suspicious Activity Detection.”

Submitted by

## Digambar Kauthkar Exam No: B150544238

## Vijay Bansode Exam No: B150544205

## Snehal Pingle Exam No: B150544258

## Pooja Idalkante Exam No: B150544224

are bonafide students of this institute and the work has been carried out by them under the supervision of **Ms. Sunita Vani** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

## Ms. Sunita Vani Mrs.V.Y.Baviskar.

## Projrct Guide Project Coordinator

**Dr. Rachana Sable**

**HOD Director**

Place: Pune Date:

i

# ACKNOWLEDGEMENT

For gaining something which is essential, there is a lot of hard work and determination that is required. For providing our group with various essentialities, we will always be thankful to the Management of “G.H. RAISONI COLLEGE OF ENGINEERING & TECHNOLOGY”.

There will always be our special expression of Gratitude towards our Head of Department Mrs. Rachana Sable for being with us in every flavor of our life, not only during this year, but all the three years that we have been together.

In aiding our group to successfully complete the project and guiding us in every possible manner, we are grateful to Mrs. Sunita Vani. It is firmly believed by us that if her presence was not there it would have been next to impossible to bring in a spark in our project.

It has been a great input from all the teaching faculties for getting till where we are now. This unconditional support by all our staff members will never be forgotten, and with due respect our group thanks them.

# NAME OF THE STUDENTS

Digambar Kauthkar

Vijay Bansode

Snehal Pingle

Pooja Idalkanthe

ii

# ABSTRACT

Suspicious Activity is predicting the body part or joint locations of a person from an image or a video. This project will entail detecting suspicious human Activity from real-time CCTV footage using neu ral networks. Human Suspicious Activity is one of the key problems in computer vision that has been studied for more than 15 years. It is important because of the sheer number of applications which can benefit from Activity detection.

For example, human pose estimation is used in applications including video surveillance, animal tracking and behavior understanding, sign language detection, advanced human-computer interaction, and marker less motion capturing. Low cost depth sensors have limitations like limited to in- door use, and their low resolution and noisy depth information make it difficult to estimate human poses from depth images. Hence, we plan to use neural networks to overcome these problems.

Suspicious human activity recognition from surveillance video is an active research area of image processing and computer vision. Through the visual surveillance, human activities can be monitored in sensitive and public areas such as bus stations, railway stations, airports, banks, shopping malls, school and colleges, parking lots, roads, etc. to prevent terrorism, theft, accidents and illegal parking, vandalism, fighting, chain snatching, crime and other suspicious activities. It is very difficult to watch public places continuously, therefore an intelligent video surveillance is required that can monitor the human activities in real-time and categorize them as usual and unusual activities; and can generate an alert. Mostly, of the research being carried out is on images and not videos. Also, none of the papers published tries to use CNNs to detect suspicious activities.

iii

# TABLE OF CONTENTS

## CHAPTER TITLE PAGE NO.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.**  1 | | **Introduction** | **Title of Chapter** | **Pg. No.**  1 |
|  | | Motivation |  | 1 |
|  | | Problem Definition |  | 2 |
| 2 |  | **Literature Survey** | | 3 |
| 3 |  | **Software Requirements Specification** | | 5 |
|  |  | 3.1.1 Software Requirement | | 5 |
|  |  | 3.1.2 User Classes and Characteristics | | 6 |
|  |  | 3.1.3 Assumptions and Dependencies | | 6 |
|  |  | **Functional Requirements** | | 7 |
|  |  | 3.2.1 Registration of Users | | 7 |
|  |  | 3.2.2 Login Implementation | | 7 |
|  |  | **Nonfunctional Requirements** | | 8 |
|  |  | 3.3.1 Portability | | 8 |
|  |  | 3.3.2 Reliability | | 8 |
|  |  | 3.3.3 Availability | | 8 |
|  |  | 3.3.4 Maintainability | | 8 |
|  |  | **System Requirements** | | 9 |
|  |  | 3.4.1 Database Requirements | | 9 |
|  |  | 3.4.2 Software Requirements | | 10 |
|  |  | 3.4.3 Hardware Requirements | | 14 |
|  |  | 3.4.4 Rapid Application Development Model (RAD) | | 14 |
| 4 |  | **System Design** | | 16 |
|  |  | System Architecture | | 16 |
|  |  | Flowchart Diagrams | | 18 |
|  |  | Use Case Diagrams | | 19 |
|  |  | Database Schema Diagram | | 23 |
|  |  | System Implementation Plan | | 26 |
|  |  | Mathematical Model | | 27 |
| 5 |  | **Other Specification** | | 28 |
|  |  | Advantages | | 28 |
|  |  | Limitations | | 28 |
|  |  | Applications | | 28 |
| 6 |  | **Conclusions & Future Work** | | 29 |
| 7 |  | **Results** | | 30 |
| 8 |  | **Reference** | | 36 |

**List of Figure**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Contents** | **Page No** |
| [3.1](#_bookmark24) | [Waterfall Model](#_bookmark24) | 20 |
| [4.1](#_bookmark28) | [System Architecture](#_bookmark28) | 23 |
| [4.2](#_bookmark30) | [Data Flow diagram](#_bookmark30) | 25 |
| [4.3](#_bookmark31) | [Data Flow diagram](#_bookmark31) | 25 |
| [4.4](#_bookmark32) | [Data Flow diagram](#_bookmark32) | 26 |
| [4.5](#_bookmark34) | [Use case Diagram](#_bookmark34) | 28 |
| [4.6](#_bookmark35) | [Activity Diagram](#_bookmark35) | 29 |
| [4.7](#_bookmark36) | [Class Diagram](#_bookmark36) | 30 |
| [4.8](#_bookmark37) | [Sequence Diagram](#_bookmark37) | 31 |
| [A.1](#_bookmark46) | [P Problem](#_bookmark46) | 37 |
| [A.2](#_bookmark47) | [NP Problem](#_bookmark47) | 38 |
| [A.3](#_bookmark48) | [NP Complete Problem](#_bookmark48) | 39 |

v

# LIST OF TABLES

**TABLE ILLUSTRATION PAGE NO.**

* 1. System Implementation Plan 26

# INTRODUCTION

We plan to build an application for detection of suspecious activity of people in public places in real time. Our application can be used in surveillance at places like malls, airports, railway stations, etc. where there is a risk of robbery or a shooting attack. We will be using deep learning and neural networks to train our system. This model will then be deployed as a mobile and desktop app which will take real time CCTV footage as input and send an alert on the administrator’s device if some suspicious pose is found. Human suspecious activity is related to identifying human body parts and possibly tracking their movements. Real life applications of it vary from gaming to AR/VR, to healthcare and gesture recognition. Compared to image data domain, there is relatively little work on applying CNNs to video classification. This is because, a video is more complex than images since it has another dimension temporal.

Unsupervised learning exploits temporal dependencies between frames and has proven successful for video analysis. Some suspecious activity approaches use CPU instead of GPU so that suspecious activity can run on low cost hardware like embedded systems and mobile phones. Low cost depth sensors are another new technology in computer vision. They are present in gaming consoles like the Kinect for Xbox 360. They are motion sensors which allow the user to interact with the console without a game controller, through just hand gestures. These are RGB-D sensors that obtain depth information by structured light technology. The structured light sensors infer the depth values by projecting an infrared light pattern onto a scene and analyzing the distortion of the projected light pattern. However, these sensors are limited to indoor use, and their low resolution and noisy depth information make it difficult to estimate human poses from depth images.

## Motivation

Human suspecious activity is one of the key problems in computer vision that has been studied for more than 15 years. It is important because of the sheer number of applications which can benefit from suspecious activity. For example, human suspecious activity is used in applications including video surveillance, animal tracking and behavior understanding, sign language detection, advanced human-computer interaction, and marker less motion capturing. Low cost depth sensors have limitations like limited to indoor use, and their low resolution and noisy depth information make it difficult to estimate human poses from depth images. Hence, we plan to use neural networks to overcome these problems.

Suspicious human activity recognition from surveillance video is an active research area of image processing and computer vision. Through the visual surveillance, human activities can be monitored in sensitive and public areas such as bus stations, railway stations, airports, banks, shopping malls, school and colleges, parking lots, roads, etc. to prevent terrorism, theft, accidents and illegal parking, vandalism, fighting, chain snatching, crime and other suspicious activities.

It is very difficult to watch public places continuously, therefore an intelligent video surveillance is required that can monitor the human activities in real-time and categorize them as usual and unusual activities; and can generate an alert. Mostly, of the research being carried out is on images and not videos. Also, none of the papers published tries to use CNNs to detect suspicious activities.

## Problem Definition

Suspecious Activity is predicting the body part or joint locations of a person from an image or a video. This project will entail detecting suspecious human Activity from real-time CCTV footage using neural networks.

# LITERATURE SURVEY

In accordance with conducted literature survey of the existing Suspicious Activity Detection systems, functionalities and their technologies that has been already implemented.

**In paper [1] Real-Time suspecious Detection and Localization in Crowded Scenes. [Mohammad Sabokrou , Mahmood Fathy]**

In this paper, we propose a method for real-time suspecious detection and localization in crowded scenes. Each video is defined as a set of non-overlapping cubic patches, and is described using two local and global descriptors. These descriptors capture the video properties from different aspects. By incorporating simple and cost-effective Gaussian classifiers, we can distinguish normal activities and anomalies in videos. The local and global features are based on structure similarity between adjacent patches and the features learned in an unsupervised way, using a sparse auto encoder. Experimental results show that our algorithm is comparable to a state-of-the-art procedure on UCSD ped2 and UMN benchmarks, but even more time-efficient. The experiments confirm that our system can reliably detect and localize anomalies as soon as they happen in a video.

**In paper [2] Learning Temporal Regularity in Video Sequences. [Mahmudul Hasan Jonghyun Choi]**

Perceiving meaningful activities in a long video sequence is a challenging problem due to ambiguous definition of ‘meaningfulness’ as well as clutters in the scene. We approach this problem by learning a generative model for regular motion patterns (termed as regularity) using multiple sources with very limited supervision. Specifically, we propose two methods that are built upon the auto encoders for their ability to work with little to no supervision. We first leverage the conventional handcrafted spatiotemporal local features and learn a fully connected auto encoder on them. Second, we build a fully convolutional feed-forward auto encoder to learn both the local features and the classifiers as an end-to-end learning framework. Our model can capture the regularities from multiple datasets. We evaluate our methods in both qualitative and quantitative ways - showing the learned regularity of videos in various aspects and demonstrating competitive performance on suspecious detection datasets as an application.

**In paper [3] suspecious Detection in Video Using Predictive Convolutional Long Short-Term Memory Networks [Jefferson Ryan Medel]**

Automating the detection of anomalous events within long video sequences is challenging due to the ambiguity of how such events are defined. We approach the problem by learning generative models that can identify anomalies in videos using limited supervision. We propose end-to end trainable composite Convolutional Long Short-Term Memory (Conv-LSTM) networks that are able to predict the evolution of a video sequence from a small number of input frames. Regularity scores are derived from the reconstruction errors of a set of predictions with abnormal video sequences yielding lower regularity scores as they diverge further from the actual sequence over time. The models utilize a composite structure and examine the effects of ‘conditioning’ in learning more meaningful representations. The best model is chosen based on the reconstruction and prediction accuracies. The Conv-LSTM models are evaluated both qualitatively and quantitatively, demonstrating competitive results on suspecious detection datasets. Conv-LSTM units are shown to be an effective tool for modeling and predicting video sequences.

**In paper [4] Abnormal Event Detection in Videos using Spatiotemporal Auto encoder. [Yong Shean Chong]**

We present an efficient method for detecting anomalies in videos. Recent applications of convolutional neural networks have shown promises of convolutional layers for object detection and recognition, especially in images. How- ever, convolutional neural networks are supervised and require labels as learning signals. We propose a spatiotemporal architecture for suspecious detection in videos including crowded scenes. Our architecture includes two main components, one for spatial feature representation, and one for learning the temporal evolution of the spatial features. Experimental results on Avenue, Subway and UCSD benchmarks confirm that the detection accuracy of our method is comparable to state-of-the-art methods at a considerable speed of up to 140 fps.

**In paper [5] Unrolled Optimization with Deep Priors. [Steven Diamond Vincent Sitzmann]**

A broad class of problems at the core of computational imaging, sensing, and low-level computer vision reduces to the inverse problem of extracting latent images that follow a prior distribution, from measurements taken under a known physical image formation model. Traditionally, handcrafted priors along with iterative optimization methods have been used to solve such problems. In this paper we present unrolled optimization with deep priors, a principled framework for infusing knowledge of the image formation into deep networks that solve inverse problems in imaging, inspired by classical iterative methods. We show that instances of the framework outperform the state-of-the-art by a substantial margin for a wide variety of imaging problems, such as denoising, deblurring, and compressed sensing magnetic resonance imaging (MRI). Moreover, we conduct experiments that explain how the framework is best used and why it outperforms previous methods.

**In paper [6] A Revisit of Sparse Coding Based suspecious Detection in Stacked RNN Framework.[Weixin Luo]**

Motivated by the capability of sparse coding based suspecious detection, we propose a Temporally-coherent Sparse Coding (TSC) where we enforce similar neigh bouring frames be encoded with similar reconstruction coefficients. Then we map the TSC with a special type of stacked Recurrent Neural Network (sRNN). By taking advantage of sRNN in learning all parameters simultaneously, the nontrivial hyper-parameter selection to TSC can be avoided, meanwhile with a shallow sRNN, the reconstruction coefficients can be inferred within a forward pass, which reduces the computational cost for learning sparse coefficients. The contributions of this paper are two-fold: i) We propose a TSC, which can be mapped to a sRNN which facilitates the parameter optimization and accelerates the suspecious prediction. ii) We build a very large dataset which is even larger than the summation of all existing dataset for suspecious detection in terms of both the volume of data and the diversity of scenes.

**In Paper [7] Connections Between Nuclear-Norm and FrobeniusNorm-Based Representations. [Xi Peng, Canyi Lu, Zhang Y]**

A lot of works have shown that frobonus norm-based representation (FNR) is competitive to sparse representation and nuclear norm-based representation (NNR) in numerous tasks such as subspace clustering. Despite the success of FNR in experimental studies, less theoretical analysis is provided to understand its working mechanism. In this brief, we fill this gap by building the theoretical connections between FNR and NNR. More specially, we prove that: 1) when the dictionary can provide enough representative capacity, FNR is exactly NNR even though the data set contains the Gaussian noise, Laplacian noise, or sample-specified corruption and otherwise, FNR and NNR are two solutions on the column space of the dictionary.

**In Paper [8], A Review of Human suspecious activity from Single Image [Naimat Ullah Khan, Wanggen Wan]**

This review is focused on the most significant contributions in Human Pose Estimation methods from a single two-dimensional image. They start their study with the traditional pictorial structure, go through a discussion of the use of Deep Neural Networks that improved the human pose estimation significantly and then the most recent, more famous approach namely Stacked Hourglass. Starting from the first practical models for estimating human pose, they provide a comprehensive study of some of the most famous deep learning methods in order to provide a concise analytical review of these most influential methods.

**In Paper [9], Human Pose Estimation using Deep Structure Guided Learning [Baole Ai, Yu Zhou, Yao Yu, Sidan Du Nanjing]**

This paper presents an approach to incorporate structure knowledge into CNNs for articulated human pose estimation from a single still image. Recent re- search on pose estimation adopt CNNs as base blocks to combine with other graphical models. Different from existing methods using features from CNNs to model the tree structure, they directly use the structure pose prior to guide the learning of CNN. First, they introduce a deep CNN with effective receptive fields which capture the holistic context of the whole image. Second, limb loss is used as intermediate supervision of CNN to learn the correlations of joints. Both parts and joints features are extract the middle of neural network and then are used to guide the following network learning.

**3. SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 SOFTWARE REQUIREMENTS**

It is the first step in the development of a system. Software requirements specification lists out all the requirements stated by the user inconsistent manner. Great software can be created only from a great specification. Systems and software these days are so complex that to get on with the design before knowing what you are going to build is foolish and risky. Software documentation is also called a software requirements specification.

Software requirements specification includes the following details:-

1. **Functionality:** It addresses what software supposed to do
2. **Performance:** It addresses the speed, response timings, availability, recovery time, software function, etc.
3. **External interface:** It addresses how the software interacts with people, the system’s hardware, other hardware, and other software.
4. **Attributes:** It addresses the portability, correctness, security, reliability, maintainability, etc.
5. **Design constraints imposed on an implementation:** It addresses the required standards in effect, implementation language, policies for database Integrity, resource limits, operating environments, etc.

**3.1.****1 User Classes and characteristics**

In this system, the user must first login and register. If registration is successful, the user can use the CNN algorithm to determine suspecious Activity Detection, Artificial Intelligence domains.

**3.1.2 Assumptions and Dependencies**

Using Python language ... Input as image data

**Dependencies:**

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it’s relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

Python is a general-purpose programming language, so it can be used for many things. Python is used for web development, AI, machine learning, operating systems, mobile application development, and video games. Python is a relatively easy programming language to learn and follows an organized structure.

Python is a general purpose and high level programming language. You can use Python for developing desktop GUI applications, websites and web applications. The simple syntax rules of the programming language further makes it easier for you to keep the code base readable and application maintainable.

**3.2 FUNCTIONAL REQUIREMENTS**

**3.2.1. System Feature (Functional Requirements)**

In order to find a solution which can be used as a part of the Reg SOC system, it is necessary to allow its integration with other modules. Research on anomaly-based intrusion detection systems is the most often carried out on the preexisting data sets or in laboratory environments in which simplification concerning infrastructure, data collection or services have been applied. Due to legal and technical limitations, our solution will detect threats through the analysis of Net Flow data and headers from network protocols. In addition, in the real environment it is not possible to obtain labeled teaching and validation datasets, which forces the introduction of adaptation mechanisms already at the deployment stage. In our approach we will try to prepare a suitably scaled model on the basis of the available datasets and then to adjust it in the following steps to the existing network. The models prepared and tuned this way will later become reference models during the implementation of the anomaly detection module in the subsequent networks

**3.3 EXTERNAL REQUIREMENTS**

**3.3.1 User Interface**

When interacting with user interfaces, do users always get what they expect? For each user interface element in thousands of Desktop App, we extracted the desktop application they invoke as well as the text shown on their screen. This association allows us to detect outliers: User interface elements whose text, context or icon suggests one action, but which actually are tied to other actions.

**3.3.2 Hardware and Software Interface**

Suspecious Activity Detection is a serious threat to network-connected embedded systems, as evidenced by the continued and rapid growth of such devices, commonly referred to as the Internet of Things. Their ubiquitous use in critical applications re- quire robust protection to ensure user safety and privacy. That protection must be applied to all system aspects, extending beyond protecting the network and external interfaces. However, embedded systems, particularly edge devices, face several challenges in applying data-driven anomaly detection, including unpredictability of malware, limited tolerance to long data collection windows, and limited computing/energy resources. In this article, we utilize sub component timing information of software execution, including intrinsic software execution, instruction cache misses, and data cache misses as features, to detect anomalies based on ranges, multi- dimensional Euclidean distance, and classification at run time. Detection methods based on lumped timing range are also evaluated and compared.

**3.4 NON FUNCTIONAL REQUIREMENTS**

**3.4.1 Performance Requirements**

In order to meet stringent performance requirements, system administrators must effectively detect undesirable performance behaviors, identify potential root causes, and take adequate corrective measures. The problem of uncovering and understanding performance anomalies and their causes (bottlenecks) in different system and application domains is well studied. In order to assess progress, research trends, and identify open challenges, we have reviewed major contributions in the area and present our findings in this survey. Our approach provides an overview of anomaly detection and bottleneck identification research as it relates to the performance of computing systems. By identifying fundamental elements of the problem, we are able to categorize existing solutions based on multiple factors such as the detection goals, nature of applications and systems, system observability, and detection meth ods.

**3.4.2 Safety Requirements**

A cumbersome task and practically infeasible in many applications. Therefore, an automated monitoring system is of both fundamental and practical interest.an intelligent solution that uses live camera images to detect workers who breach safety rules by not wearing high-visibility vests. The proposed solution is formulated in the form of an anomaly detection algorithm developed in the random finite set (RFS) frame- work.

**3.4.3 Security Requirements**

Suspecious Activity Detection is the process of finding outliers in a given dataset. Outliers are the data objects that stand out amongst other objects in the dataset and do not conform to the normal behavior in a dataset. Anomaly detection is a data science application that combines multiple data science tasks like classification, re aggression, and clustering. The target variable to be predicted is whether a transaction is an outlier or not. Since clustering tasks identify outliers as a cluster, distance-based and density-based clustering techniques can be used in anomaly detection tasks.

**3.4.4 Software Quality Attributes.**

Software has many quality attribute that are given below:-

* Adaptability: This software is adaptable by all users.
* Availability: This software is freely available to all users. The availability of the software is easy for everyone.
* Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
* Reliability: The performance of the software is better which will increase the reliability of the Software.
* User Friendliness: Since, the software is a GUI application; the output generated is much user friendly in its behavior.
* Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.
* Security: Users are authenticated using many security phases so reliable Security is provided.
* Testability: The software will be tested considering all the aspects.

**3.5 SYSTEM REQUIRENETS**

**3.5.1 Database Requirement**

The Database Requirements involves the use of a lot of information, some which will be needed several times and the most appropriate form of storage of this data is in a database. This will allow data to be saved from input to the Database Requirements and retrieved to be used by the Database Requirements.

As an important aspect of this project is use of Time Control System. In this section several databases are reviewed for their suitability to this project.

**3.5.2 Software Requirements.**

RAM: 8 GB

Processor: Intel i5 Processor

IDE: Spyder

Coding Language: Python Version 3.8

Operating System: Windows 10

**3.5.3 Hardware Requirements**

Speed: 1.1 GHz

Hard Disk: 40 GB

Key Board: Standard Windows

Keyboard Mouse: Two or Three Button Mouse

Monitor: LCD/LED

**3.6 ANYALYSIS MODELS:**

**3.6.1 SDLC MODLE APPLIED**

**Software Development Life Cycle (SDLC)**. SDLC is a popular practice that is followed by different organizations for designing and developing high-quality software applications.  It acts as a framework that holds some specific tasks to be achieved at every phase during the software development progression. This article will give you deep insight into the need for software development in various stages of SDLC.

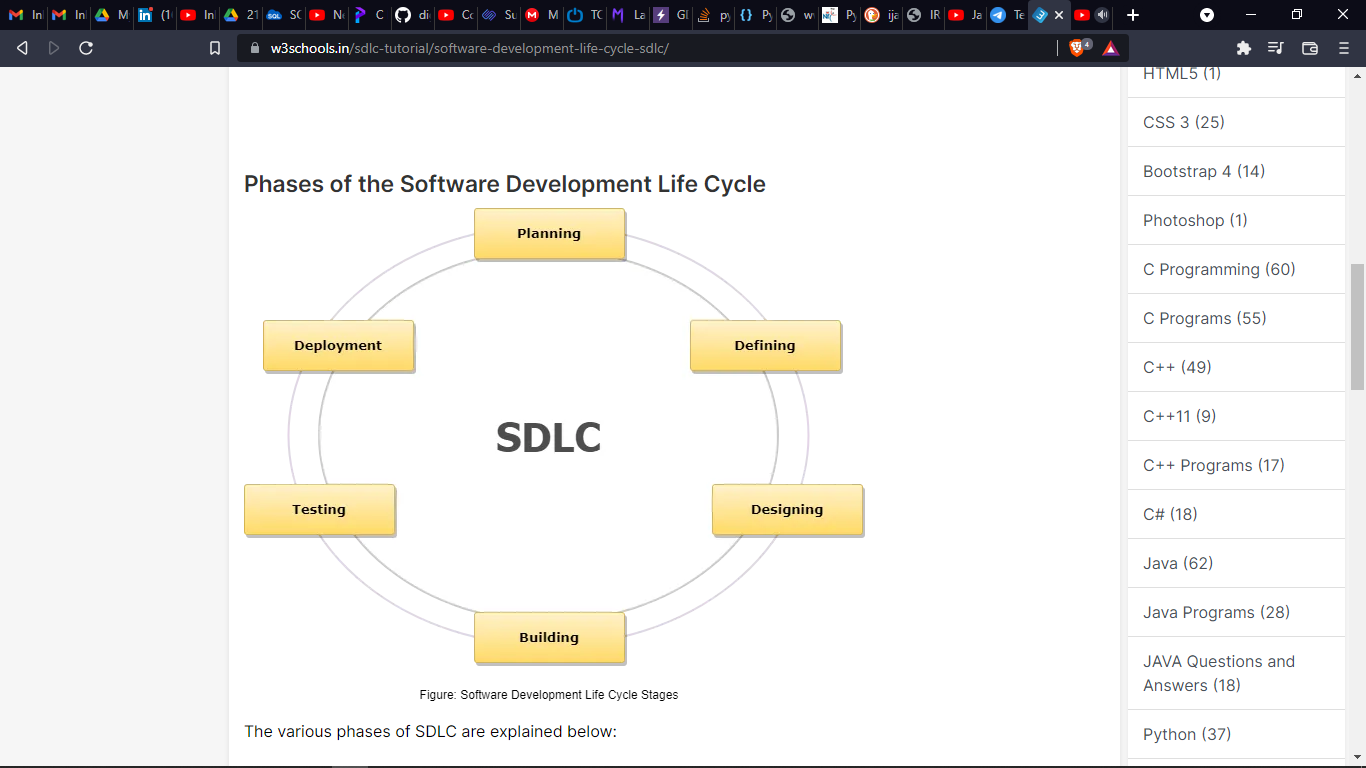


Fig .3.6.1 SDLC Life Cycle.

The various phases of SDLC are explained below:

* **First Phase: Requirement Collection or Planning Phase**

The prime focus of this phase is to gather the essential requirements from the customer. This information gets collected by the business analyst from their target customer(s) and plans the BRS (Business requirement Specification) for the development of the product. The team of all the designers and BA will do brainstorming to extract all the requirements and plan accordingly for the new system to be developed?

* **Second Phase: Defining or Feasibility Study Phase**

When the BRS documentation is done, there are another set of employees like Human Resource (HR), Finance Analyst, Architect, a Business analyst as well as Project manager will sit jointly discuss as well as analyze how to proceed and whether it is feasible and possible in the allotted budget. Such decisions are taken depending on the cost, resources, time, etc. Documentation is made, which is the SRS (Software Requirement Specification) document, which contains a detailed explanation of product requirements, right from design to development.

* **Third Phase: Designing Phase**

This phase is when the design specification is organized from the prerequisite document when the project is approved to go further. This phase contributes to the next phase for development. This phase portrays a blueprint of the product, which helps to specify the hardware and requirements of your system as well as assist in crafting a significant architecture of your system.

* **Fourth Phase: Building or Coding Phase**

As you are preparing with the design document, this phase deals with the developers to start writing the code or prepare for the engineering so that a prototype of the product can be created using some specific tools and techniques. This is considered the longest phase of SDLC.

* **Fifth Phase: Testing Phase**

As your product is prepared for deployment, it needs a prior testing environment by the test engineers to check for bugs and run-time errors, and they check in this phase whether the functionality of the product is working as per the requirement or not. The bugs or defects which are encountered in the test phase are reported to the developers, who fix the bug and revert to the test engineers for further testing. This is an iterative process that continues until your application is free from bugs and defects and works stably.

* **Sixth Phase: Deployment Phase**

Once your prototype or product is developed, tested, and completely in working form as per the requirement, and then it is installed or deployed in the customer's workplace or system for their use.

* **Seventh Phase: Maintenance Phase**

This is an additional phase, and in many cases, **this phase does not come under the count of SDLC**, when your customer(s) begin using your product and encounter with some issues which they want us (as developers) to fix from time to time. The developer fixes the issue, and software testers test the product and hand it over the back to the customer.

**3.6.2 Water Fall Model**

The classical waterfall model which is also known as the linear-sequential life cycle model is an essential software development model which can be understandable from the structure itself. The model is straightforward yet idealistic. When this model was first introduced, it used to be very popular, but time, the new model has come up with a change in features and requirements and hence it is used decidedly less but still a popular one which everyone must know. All the old software has been developed based on this model's life cycle. It is a sequential model which segregates software development into different phases. Each phase is designed with some unique functionality and use. The model was pioneered in the year 1970 by Winston Royce.

The various phases of the Waterfall Model which are explained below:

1. Requirement Gathering Stage/Feasibility Study
2. Design Stage
3. Built Stage
4. Integration and Test Stage
5. Deployment Stage
6. Maintenance Stage

The different chronological phases of the waterfall model are shown below with the interconnection between them:

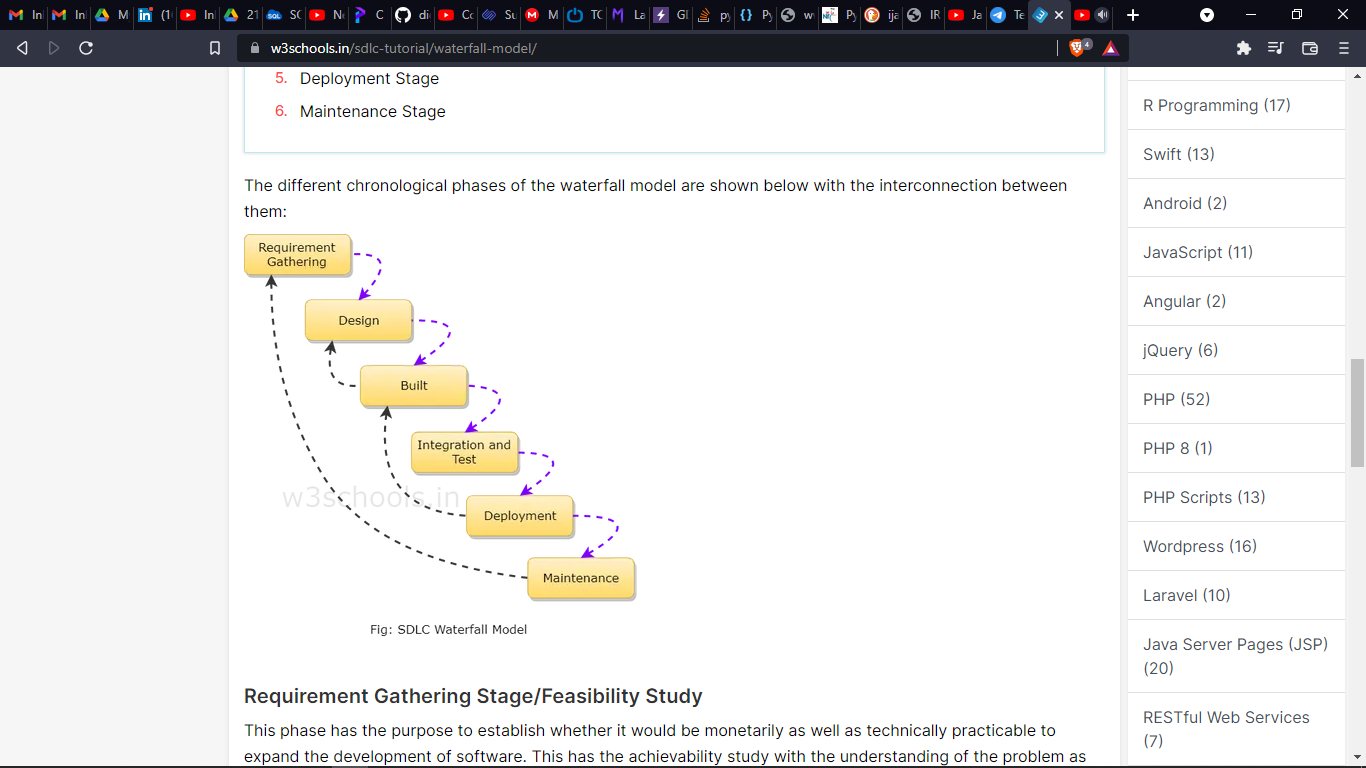


Fig.3.6.2 Waterfall Model.

* **Requirement Gathering Stage/Feasibility Study**

This phase has the purpose to establish whether it would be monetarily as well as technically practicable to expand the development of software. This has the achievability study with the understanding of the problem as well as determines the diverse potential strategies used for solving the problem.

* **Design Stage**

There is a thorough study of the entire requirement specifications from the first phase, and then the system design is equipped. This phase helps developers to specify hardware as well as the system's requirement which ultimately helps in characterizing the system design as a whole.

* **Built Stage**

This phase is also known as the coding phase of software development where the idea is converted into source code and UI plus UX design using programming language and tools. Hence, every designed module needs to be coded.

* **Integration and Test Stage**

Once the coding of application is done, it is then integrated with all other modules with different functionality. During each step of integration, earlier planned modules are incorporated into the parts included the structure of the software and then the entire system is tested.

**α Testing**: In this testing, the software is tested by the development team, i.e., the developers.

**β Testing**: In this testing, the software is tested by friendly customers and other target users who will use the beta version of your product.

**Acceptance Testing**: Once the application has been distributed, the customer carries out the acceptance test for determining if the product should be accepted as delivered or rejects it for further modification.

* **Deployment Stage**

As all the functional, as well as non-functional tests, are completed, the software is installed in the customer's end or the environment or gets released in the market.

* **Maintenance Stage**

Another important phase of this model is the maintenance model. Updating the product, patching any bugs and errors and developing other essential components as per feedback to make this full software is done in this stage. It is of three types:

**Corrective Maintenance**: Corrective maintenance is where the maintenance is done to fix the errors.

**Perfective Maintenance**: Perfective maintenance is done where the maintenance is done to increase the efficiency of any system according to customer's requirement.

**Adaptive Maintenance**: Adaptive Maintenance is typically necessary for porting your application to a new work environment or porting from one type of OS to another.

**4. SYSTEM DESIGN**

**4.1 System Architecture**

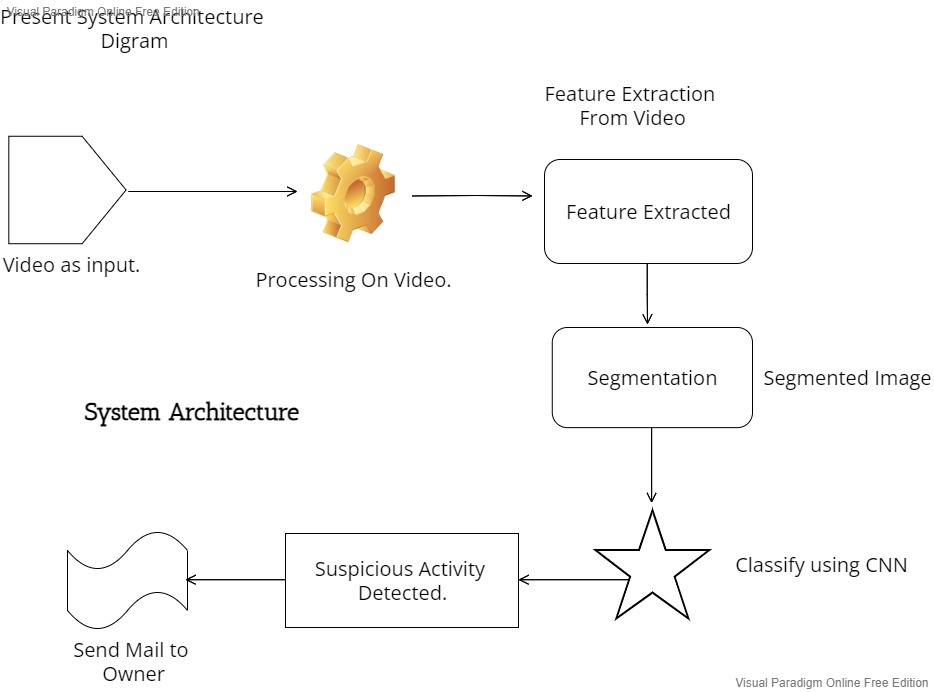


Fig4.1 .System Architecture

**4.2 Data Flow Diagram (DFD)**

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

**Data Flow Diagram Levels**

Context Diagram. A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities.

**DFD Layers.** Draw data flow diagrams can be made in several nested layers. A single process node on a high level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.

**DFD Levels.** The first level DFD shows the main processes within the system. Each of these processes can be broken into further processes until you reach pseudo code.

**0-Level DFD**

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles.

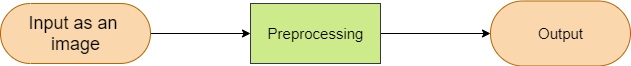


Fig.4.2.1 0-Level DFD

**1-Level DFD**

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into sub processes.

Fig.4.2.2 1-Level DFD



**2-Level DFD**

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.



Fig.4.2.3 2-Level DFD

**4.3 UML Diagrams**

The UML stands for Unified modeling language, is a standardized general-purpose visual modeling language in the field of Software Engineering. It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system. It helps in designing and characterizing, especially those software systems that incorporate the concept of Object orientation. It describes the working of both the software and hardware systems.

The UML has the following features:

1. It is a generalized modeling language.
2. It is distinct from other programming languages like C++, Python, etc.
3. It is interrelated to object-oriented analysis and design.

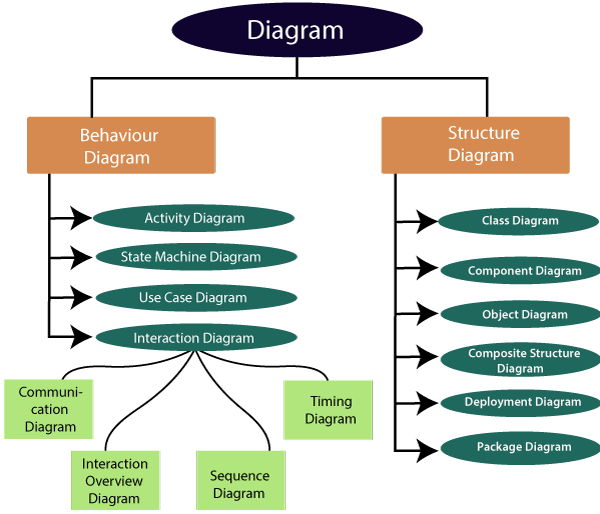


Fig.4.3.1 Classification of UML Diagrams

**4.3.1 Use Case Diagrams**

It represents the functionality of a system by utilizing actors and use cases. It encapsulates the functional requirement of a system and its association with actors. It portrays the use case view of a system.

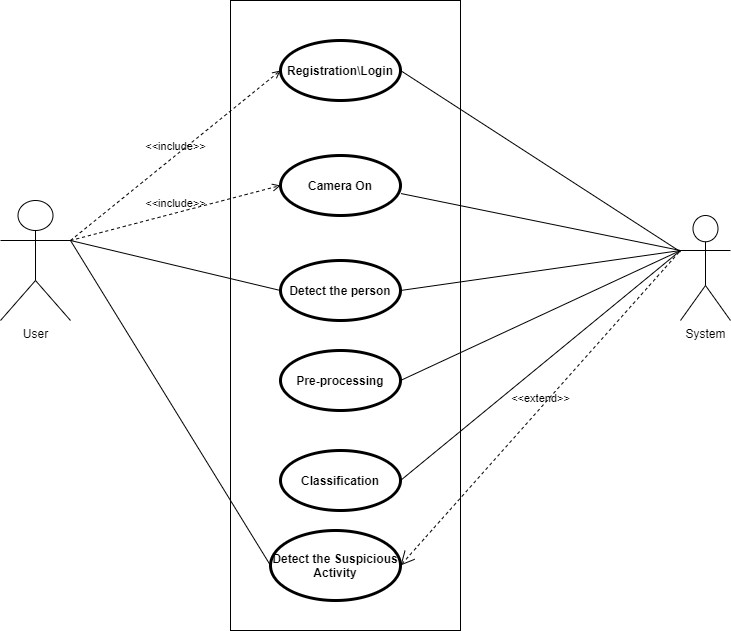
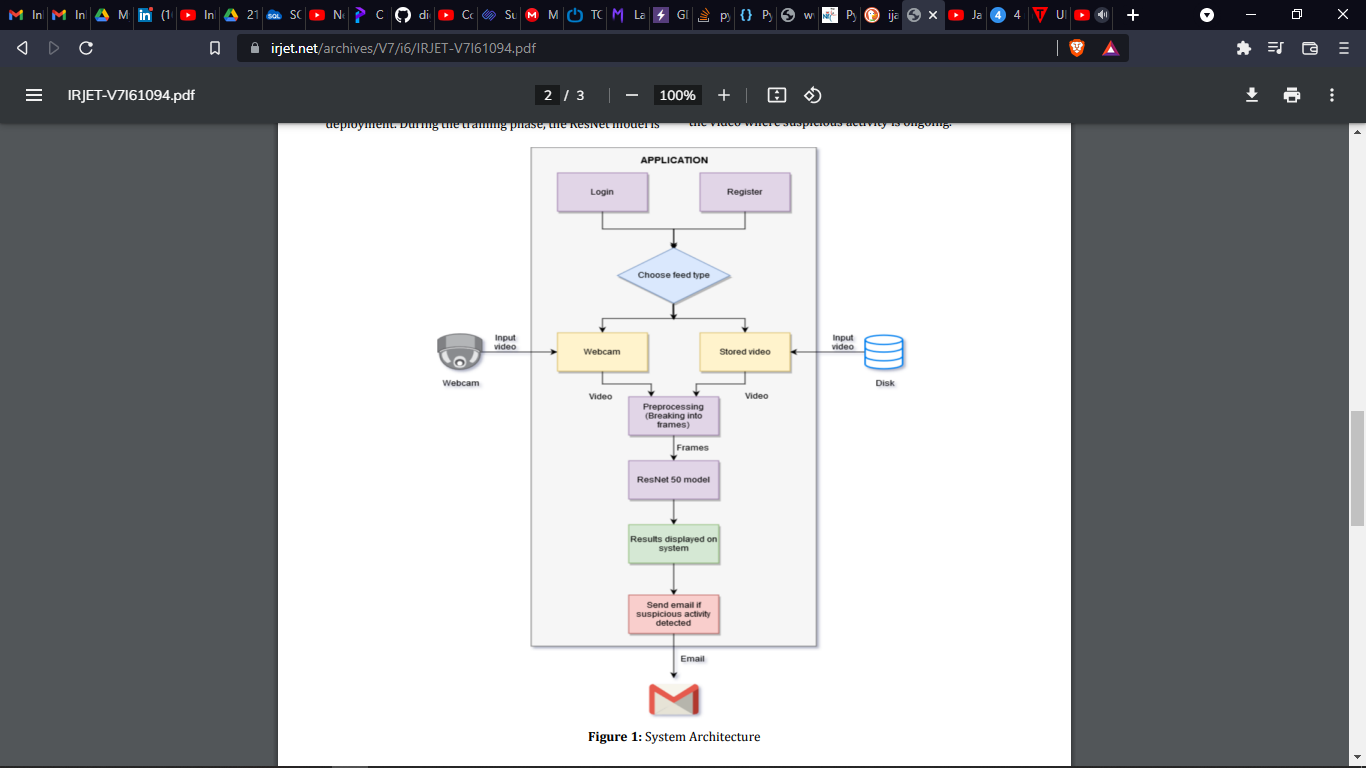


Fig.4.3.2 Use Case Diagram

**4.3.2 Activity Diagram**

It models the flow of control from one activity to the other. With the help of an activity diagram, we can model sequential and concurrent activities. It visually depicts the workflow as well as what causes an event to occur.



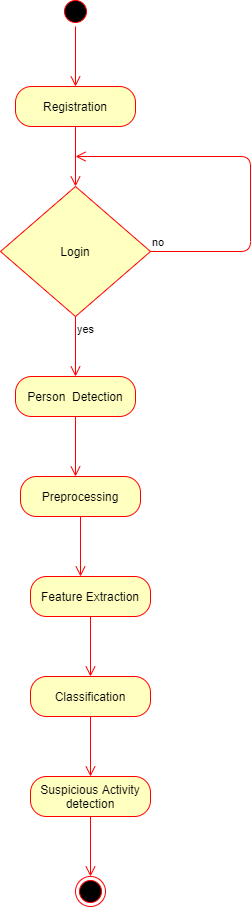


Fig.4.3.3 Activity Diagram

**4.3.3 Class Diagram**

Class diagrams are one of the most widely used diagrams. It is the backbone of all the object-oriented software systems. It depicts the static structure of the system. It displays the system's class, attributes, and methods. It is helpful in recognizing the relation between different objects as well as classes.

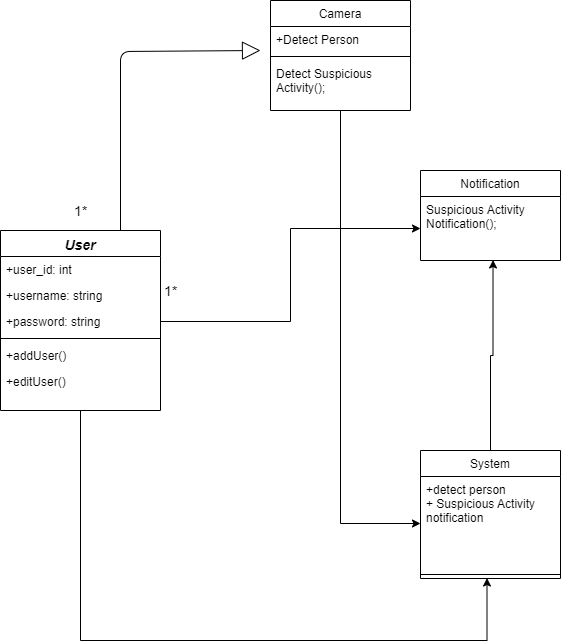


Fig.4.3.4 Class Diagram

**4.3.4 Sequence Diagram**

It shows the interactions between the objects in terms of messages exchanged over time. It delineates in what order and how the object functions are in a system.

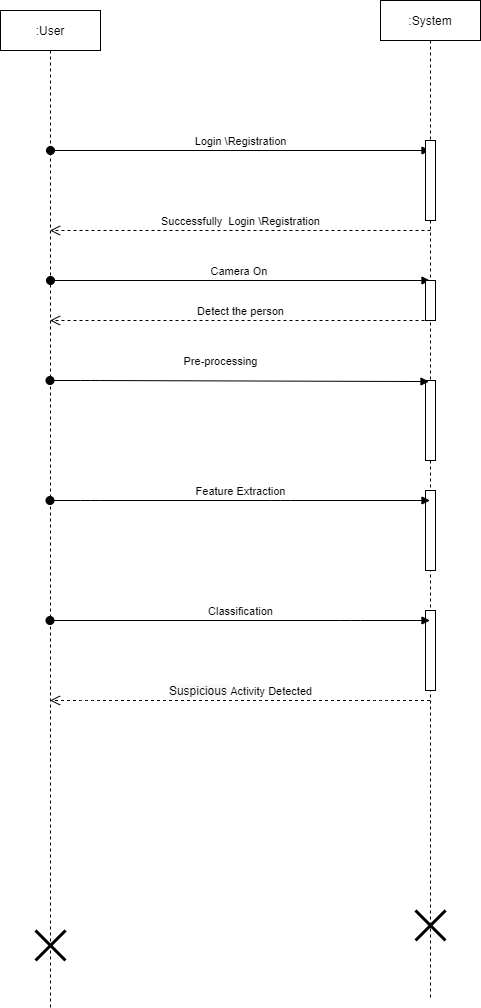


Fig.4.3.5 Sequence Diagram

**6. OTHER SPECIFICATIONS**

**6.1 Advantages**

1. Easily detect the suspecious activity.
2. Decrease the number of crime.
   1. **Limitations**
3. If the training not get successful or get interrupt because of any reason then system cannot work proper.
4. If the accuracy of training less then system cannot work properly.

**6.3 Applications**

1. Crowd Area
2. Hospital
3. Offices
4. Roads

# Conclusions & Future Work

**7.1 Conclusion**

A system to process real-time CCTV footage to detect any suspecious activity will help to create better security and less human intervention. Great strides have been made in the field of human suspecious Activity, which enables us to better serve the myriad applications that are possible with it. Moreover, research in related fields such as Activity Tracking can greatly enhance its productive utilization in several fields.

**7.2 Future Scope**

We implement this system on android.

Also we can try to improve accuracy.