### SAVITRIBAI PHULE PUNE UNIVERSITY

**A PRELIMINARY PROJECT REPORT ON**

## Mobile Botnet Detection A Machine Learning Approach using SVM

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

**BACHELOR OF ENGINEERING(Computer Engineering) BY**

Milind Bramhane B190984203

Madhuri Bhusare B190984202

Dhananjay Nimse B190984218

Vaibhav Sangle B190984225

**Under The Guidance of**

Prof. S.Y. Kale



**DEPARTMENT OF COMPUTER ENGINEERING**

**Adsul’s Technical Campus**

**Chas, Ahmednagar**



**Adsul’s Technical Campus**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CERTIFICATE**

This is to certify that the Project Entitled

## Mobile Botnet Detection A Machine Learning Approach using SVM

Submitted by

Milind Bramhane B190984203

Madhuri Bhusare B190984202

Dhananjay Nimse B190984218

Vaibhav Sangle B190984225

is a bonafide work carried out by Students under the supervision of Prof. S.Y. Kale Name and it is submitted towards the partial fulfillment of the requirement of Bach- elor of Engineering (Computer Engineering) Project.

Prof. S.Y. Kale Prof. J. R. Mahajan

Internal Guide Project Coordinator

Dept. of Computer Engg. Dept. of Computer Engg.

Dr. H. B. Jadhav Dr. P. M. Patil

H.O.D. Principal

Dept. of Computer Engg. ATC, A’nagar

Date:

Place: Ahmednagar

Savitribai Phule Pune University



CERTIFICATE

This is to certify that

Milind Bramhane B190984203

Madhuri Bhusare B190984202

Dhananjay Nimse B190984218

Vaibhav Sangle B190984225

Student of B.E. Computer Engineering was examined

Project Stage - II entitled

## Mobile Botnet Detection A Machine Learning Approach using SVM

Towards the partial fulfillment of

Undergraduate’s degree in Computer Engineering during the

Year 2022-23

Internal Examiner External Examiner

**Abstract**

**Botnets have been a serious threat to the Internet security. With the constant sophistication and the resilience of them, a new trend has emerged, shifting botnets from the traditional desktop to the mobile environment. As in the desktop domain, detecting mobile botnets isessential to minimize the threat that they impose. Along the diverse set of strategies applied to detect these botnets, the ones that show the best and most generalized results involve dis- covering patterns in their anomalous behavior. In the mobile botnet field, one way to detect these patterns is by analyzing the operation parameters of this kind of applications. In this paper, we present an anomaly-based and host-based approach to detect mobile botnets. The proposed approach uses machine learning algorithms to identify anomalous behaviors in statistical features extracted from system calls. We were able to test the performance of our approach in a close-to-reality scenario. The proposed approach achieved great results, including low false positive rates and high.**

**Acknowledgments**

It gives us great pleasure in presenting the preliminary project report on **Mobile Botnet Detection A Machine Learning Approach using SVM**

I would like to take this opportunity to thank my internal guide **Prof. S.Y. Kale** for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

We are also grateful to **Dr. Jadhav H.B** Head of Computer Engineering Department, ATC for his indispensable support and suggestions.

In the end our special thanks to **Prof. Mahajan J.R** for providing various resources for our Project.

Milind Bramhane

Madhuri Bhusare

Dhananjay Nimse

Vaibhav Sangle

**INDEX**

1. [Introduction](#_bookmark0) 1

[1.1 Introduction](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

[1.2 Motivation](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

[1.3 Problem Statement](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1. [Literature Survey](#_bookmark4) 4
2. [Software Requirements Specification](#_bookmark5) 10
   1. [Introduction](#_bookmark6) 11
      1. [Algorithm Support Vector Machine](#_bookmark7) 11
      2. [Project Scope](#_bookmark8) 12
      3. [Design And Implementation Constraints](#_bookmark9) 13
   2. [External Interface Requirements (If Any)](#_bookmark10) 13
      1. [User Interface](#_bookmark11) 13
      2. [Hardware Interfaces:](#_bookmark12) 13
      3. [Software Interfaces](#_bookmark13) 13
   3. [Research And Development Challenges](#_bookmark14) 14
      1. [Technical challenges](#_bookmark15) 14
   4. [Functional Requirements](#_bookmark16) 15
      1. [System Feature (Functional Requirement)](#_bookmark17) 15
   5. [NON FUNCTIONAL REQUIREMENT](#_bookmark18) 16
      1. [Performance Requirements](#_bookmark19) 16
      2. [Safety Requirement](#_bookmark20) 16
      3. [Software Quality Attributes](#_bookmark21) 17
   6. [System Requirements](#_bookmark22) 18
      1. [Database Requirements](#_bookmark23) 18
      2. [Software Requirements](#_bookmark24) 18
      3. [Hardware Requirements](#_bookmark25) 19
   7. [Analysis Models: SDLC Model to be applied](#_bookmark26) 20
   8. [System Implementation Plan](#_bookmark27) 22
3. [Project Plan](#_bookmark28) 23
   1. [Reconciled Estimates](#_bookmark29) 24
      1. [Project Resource](#_bookmark31) 25
   2. [Risk Management](#_bookmark32) 26
      1. [Risk Identification](#_bookmark33) 26
      2. [Risk Analysis](#_bookmark34) 27
      3. [Risk Mitigation Risk Monitoring and Risk Management](#_bookmark35) 28
   3. [Project Schedule](#_bookmark37) 29
      1. [Project Task Set](#_bookmark38) 29
      2. [Task Network](#_bookmark39) 30
      3. [Timeline Chart](#_bookmark41) 30
   4. [Team Organization](#_bookmark43) 31
      1. [Team Structure](#_bookmark44) 31
      2. [Management Reporting And Communication](#_bookmark45) 31
4. [System design](#_bookmark46) 32
   1. [System Architecture](#_bookmark47) 33
   2. [Data Flow Diagrams](#_bookmark49) 34
      1. [Data Flow Diagram](#_bookmark50) 34
5. [Software Testing](#_bookmark58) 40
   1. [Purpose](#_bookmark59) 41
   2. [Scope of testing](#_bookmark60) 42
   3. [Software to be tested](#_bookmark61) 43
   4. [Test Plan](#_bookmark62) 44
   5. [Type of testing](#_bookmark63) 45
      1. [Unit Testing](#_bookmark64) 45
      2. [Regression Testing](#_bookmark65) 46
      3. [Smoke Testing](#_bookmark66) 46
      4. [System Testing](#_bookmark67) 46
6. [TECHNICAL SPECIFICATION](#_bookmark68) 47
   1. [Advantages](#_bookmark69) 48
   2. [Disadvantages:](#_bookmark70) 48
   3. [Application](#_bookmark71) 48
7. [Test Cases](#_bookmark72) 49
8. [Result](#_bookmark77) 52
9. [Conclusions](#_bookmark84) 56
   1. [Future Work](#_bookmark85) 57
10. [References](#_bookmark86) 58

**List of Figures**

[4.1](#_bookmark30) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

[4.2](#_bookmark36) Risk Mitigation Risk Monitoring And Risk Management 28

* 1. [Task Network](#_bookmark40) 30
  2. [Timeline Chart](#_bookmark42) 30
  3. [System Architecture](#_bookmark48) 33
  4. [Data Flow0 diagram](#_bookmark51) 34
  5. [Data Flow 1diagram](#_bookmark52) 34
  6. [Data Flow2 diagram](#_bookmark53) 35
  7. Usecase Diagram 36
  8. [Activity Diagram](#_bookmark55) 37
  9. [Sequence Diagram](#_bookmark56) 38
  10. [Class Diagram](#_bookmark57) 39
  11. [GUI Testing](#_bookmark73) 50
  12. [login test Cases](#_bookmark74) 50
  13. [Registration Test cases](#_bookmark75) 51
  14. [System Test Cases](#_bookmark76) 51
  15. [main page](#_bookmark78) 53
  16. [registration page](#_bookmark79) 53
  17. [login page](#_bookmark80) 54
  18. [output1](#_bookmark81) 54
  19. [output2](#_bookmark82) 55
  20. [output3](#_bookmark83) 55

# CHAPTER 1 INTRODUCTION

## INTRODUCTION

Android is now the most widespread mobile operating system worldwide. Over the years the volume of malware targeting Android has continued to grow. This is because it is easier and more profitable for malware authors to target an operating sys- tem that is open-source, more prevalent, and does not restrict the installation of apps from any possible source. As a matter of fact, numerous families of malware apps that are capable of infecting Android devices and turning them into malicious bots have been discovered in the wild. These Android bots may become part of a larger botnet that can be used to perform various types of attacks such as Distributed Denial of Service (DDoS) attacks, generation and distribution of Spam, Phishing attacks, click fraud, stealing login credentials or credit card details, etc. A botnet consists of a number of Internet-connected devices under the control of a malicious user or group of users known as bot masters. It also consists of a Command and Control (CC) infrastructure that enables the bots to receive commands, get updates and send status information to the malicious actors. Since smartphones and other mobile devices are typically used to connect to online services and are rarely switched off, they provide a rich source of candidates for operating botnets. Thus, the term ‘mobile botnet’ refers to a group of compromised smartphones and other mobile devices that are remotely controlled by bot masters using CC channels. Nowadays, malicious botnet apps have become a serious threat. Additionally, their increasing use of sophisticated evasive techniques calls for more effective detection approaches. Hence, in this paper we present a deep learning approach that leverages Support Vector Machine (SVM) for Android botnet detection. The SVM model employs 342 static features to classify new or previously unseen apps as either ‘botnet’ or ‘normal’. The features are extracted through automated reverse engineering of the apps, and are used to create feature vectors that feed directly into the SVM model without further pre-processing or feature selection.

## MOTIVATION

1. Because, in this case, it is difficult to distinguish between botnet and benign net- work traffic.
2. Botnets can prevent themselves from being identified by generating different do- mains, so centralizing machine learning techniques to identify botnets based on the IP address of the sender and receiver no longer gives an accurate answer to security researchers

As a result of using machine learning techniques (in basic mode) and adjusting its parameters manually, it may be able to generate a good answer for a single botnet dataset, but if the same technique is implemented on different datasets with the same parameters,

## PROBLEM STATEMENT

Our proposed botnet detection system is implemented as a SVM-based model that is trained on 342 static app features to distinguish between botnet apps and normal apps.

# CHAPTER 2 LITERATURE SURVEY

1. **Paper Name:** ABIS: A Prototype of Android Botnet Identification System.

**Author:** Chanin Tansettanakorn, Supachai Thongprasit.

**Description:** According to the advanced wireless technology in nowadays, most people mainly use their mobile phones as an essential tool. At the same time, threats of the mobile phones such as virus, bot nets, and other malware are also increasing as well. However, most users have the limited of knowledge about mobile threats.

1. **Paper Name:** Android Botnet Classification Using Feature Selection and Classification Algorithms.

**Author:** Zubaile Abdullah.

**Description:** Smartphones have become an important part of human lives, and this led to an increase number of smartphone users. However, this also attracts hackers to develop malicious applications especially Android botnet to steal the private information and causing financial losses. Due to the fast modifications in the technologies used by malicious application (app) developers.

1. **Paper Name:** Toward a Detection Framework for Android Botnet.

**Author:** Wadi ijawi, Ja’far Alqatawna Hossam Faris

**Description:** Android is one of the most popular and widespread operating systems for smartphones. It has several millions of applications that are published at either official or unofficial stores. Botnet applications are kind of malware that can be published using these stores and downloaded by the victims on their smartphones.

1. **Paper Name:** Mobile Botnet Detection: A Deep Learning Approach Using Convolutional Neural Networks

**author name:** Suleiman Y. Yerima1 and Mohammed K. Alzaylaee2

**Description:** being the most widespread mobile operating systems is increasingly becoming a target for malware. Malicious apps designed to turn mobile devices into bots that may form part of a larger botnet have become quite common, thus posing a serious threat. This calls for more effective methods to detect botnets on the Android platform. Hence, in this paper, we present a deep learning approach for Android bot- net detection based on Convolutional Neural Networks (CNN). Our proposed botnet detection system is implemented as a CNN-based model that is trained on 342 static app features to distinguish between botnet apps and normal apps.

1. **Paper Name:** Cooperative Network Behavior Analysis Model for Mobile Botnet Detection

**author name:** Meisam Eslahi1, Moslem Yousefi2, Maryam Var Naseri

**Description:** Recently, the mobile devices are well integrated with Internet and widely used by normal users and organizations which employ Bring Your Own Device Technology On the other hand, the mobile devices are less protected in comparison to computers. Therefore, the mobile devices and networks have now become attractive targets for attackers. Amongst several types of mobile threats, the mobile HTTP Botnets can be considered as one of the most sophisticated attacks. A HTTP Bots stealthily infect mobile devices and periodically communicate with their controller called Botmaster.

1. **Paper Name:** Detection of Mobile Botnets using Neural Networks

**author name:** Milan Oulehla

**Description:** his poster deals with botnets, the most dangerous kind of mobile mal- ware, and their detection using neural networks. Unlike common mobile malware, botnets often have a complicated pattern of behavior because they are not managed by predictable algorithms but they are controlled by humans via command and control servers (CC servers) or via peer-to-peer networks.

1. **Paper Name:** A Static Approach towards Mobile Botnet Detection

**author name:** Shahid Anwar

**Description:** The use of mobile devices, including smartphones, tablets, smart watches and notebooks are increasing day by day in our societies. They are usually connected to the Internet and offer nearly the same functionality, same memory and same speed like a PC. To get more benefits from these mobile devices, applications should be in- stalled in advance. These applications are available from third party websites, such as google play store etc. In existing mobile devices operating systems, Android is very easy to attack because of its open source environment. Android OS use of open source facility attracts malware developers to target mobile devices with their new malicious applications having botnet capabilities. Mobile botnet is one of the crucial threat to mobile devices. In this study we propose a static approach towards mobile botnet detection.

1. **Paper Name:** Toward a Detection Framework for Android Botnet

**author name:** Wadi’ Hijawi†, Ja’far Alqatawna†, Hossam Faris

**Description:** Android is one of the most popular and widespread operating systems for smartphones. It has several millions of applications that are published at either official or unofficial stores. Botnet applications are kind of malware that can be published using these stores and downloaded by the victims on their smartphones. In this paper, we propose Android botnet detection method based a new set of dis- criminating features extracted based from the analysis of Android permissions (i.e. Protection levels for all available Android permissions). Then we compared the pre- diction power of different machine learning models before and after adding these features to the state-of-art requested permissions features in Android.

1. **Paper Name:** Mobile Botnet Detection: Proof of Concept

**author name:** Zubaile Abdullah1,2, Madihah Mohd Saudi2 , Nor Badrul Anuar **Description:** Nowadays mobile devices such as smartphones had widely been used. People use smartphones not limited for phone calling or sending messages but also for web browsing, social networking and online banking transaction. To certain extend, all confidential information is kept in their smartphone. As a result, smart- phones became as one of the cyber-criminal main target especially through an installation of mobile botnet. Eurograbber is an example of mobile botnet that being installed via infected mobile application without victim knowledge

1. **Paper Name:** ABIS: A Prototype of Android Botnet Identification System **author name:** Chanin Tansettanakorn, Supachai Thongprasit **Description:** According to the advanced wireless technology in nowadays, most people mainly use their mobile phones as an essential tool. At the same time, threats of the mobile phones such as virus, bot nets, and other malware are also increasing as well. However, most users have the limited of knowledge about mobile threats. Therefore, we would like to reduce the number of botnet infected mobile phones before the users install an application on their phones. We develop a system called ABIS (Android Botnet Identification System) to check Android applications whether they are possibly being malware or not.

# CHAPTER 3

**SOFTWARE REQUIREMENTS SPECIFICATION**

## INTRODUCTION

## Algorithm: Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Sup- port Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

1. **Linear SVM:** The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue.
2. **Non-Linear SVM:** If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.

**S**VM Method –

1.from sklearn.svm import SVC”Supportvector classifier” 2.classifier=SVC(kernel=’linear’,random*state* = 0) 3*.classi f ier. f it*(*xtrain, ytrain*)

## Project Scope

Mobile botnets typically contain code that can be used for sending messages, attacking other networks or conducting DDoS attacks. Researchers have also discovered that mobile botnets will only infect devices running on particular versions of the An-droid operating system

## Design And Implementation Constraints

1. Collect Dataset
   1. Data PROCESSING
   2. Detect Mobile Botnet

## EXTERNAL INTERFACE REQUIREMENTS (IF ANY)

## User Interface

Application Based Mobile Botnet Detection A Machine Learning Approach using SVM.

## Hardware Interfaces:

Processor: Intel i5 and above Operating System: Windows 10 Ram: 8GB

## Software Interfaces

Programming Language: Python IDE: Spyder

Software Anaconda Navigator: Tkinter

## RESEARCH AND DEVELOPMENT CHALLENGES

## Technical challenges

1. Following are some of the technical challenges faced while developing the proposed system.
   1. **Choosing programming language:**

In our proposed system as we are using concepts of image processing and background subtraction algorithm, that’s why we decided to use PYTHON language. PYTHON is open source programming language and easy to implement.

* 1. All image formats should be supported by the system

## 3. Understanding the System Algorithms:

Background subtraction is a widely used approach for detecting moving objects from the difference between the current frame and a reference frame, often called” background image”, or” background model”. This algorithm improves the performance of the system.

## FUNCTIONAL REQUIREMENTS

## System Feature (Functional Requirement)

system feature Proposed system consists of 4 modules:

1. Feature point extraction: Feature points of each Dataset parameters gets detected.
2. Feature correspondence matching: Matching of selected feature points across various parameters.
3. Point estimation: Position estimation and vision system orientation during navigation
4. Position refinement: Location estimate based, accurate location derivation

## NON FUNCTIONAL REQUIREMENT

## Performance Requirements

The performance of the functions and every module must be well. The overall performance of the software will enable the users to work efficiently. Performance of encryption of data should be fast. Performance of the providing virtual environment should be fast Safety Requirement.

The application is designed in modules where errors can be detected and easily.

This makes It easier to install and update new functionality if required.

## Safety Requirement

The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required.

## Software Quality Attributes

Our 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.

## SYSTEM REQUIREMENTS

## Database Requirements

Browser for SQLite (DB4S) is a high quality, visual, open source tool to create, de- sign, and edit database files compatible with SQLite.

DB4S is for users and developers who want to create, search, and edit databases. DB4S uses a familiar spreadsheet-like interface, and complicated SQL commands do not have to be learned.

Controls and wizards are available for users to:

Create and compact database files Create, define, modify and delete tables Create, define, and delete indexes Browse, edit, add, and delete records Search records

Import and export records as text

Import and export tables from/to CSV files

Import and export databases from/to SQL dump files Issue SQL queries and inspect the results

## Software Requirements

Anaconda Navigator: Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command line commands. Navigator can search for packages on Anaconda.org or in a

local Anaconda Repository. It is available for Windows, and Linux. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

## Hardware Requirements

RAM: 8 GB

As we are using Machine Learning Algorithm and Various High Level Libraries Laptop

RAM minimum required is 8 GB. Hard Disk: 40 GB

Data Set of CT Scan images is to be used hence minimum 40 GB Hard Disk memories required.

Processor: Intel i5 Processor IDE: Spyder

Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of

a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

## ANALYSIS MODELS: SDLC MODEL TO BE APPLIED

SDLC Models stands for Software Development Life Cycle Models. In this article, we explore the most widely used SDLC methodologies such as Agile. Each software development life cycle model starts with the analysis, in which the Also, here are defined the technologies used in the project, team load. One of the basic notions of the software development process is SDLC models which stands for Soft- ware Development Life Cycle models. SDLC – is a continuous process, which starts from the moment, when it’s made a decision to launch the project, and it ends at the moment of its full remove from the exploitation. There is no one single SDLC model. They are divided into main groups, each with its features and weaknesses.

1. Requirement Analysis - Requirement Analysis is the most important and necessary stage in SDLC. The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry. Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage. Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.
2. System Design - The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.
3. Implementation - In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.
4. Testing - After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage. During this stage, unit testing, integration testing, system testing, acceptance testing is done.
5. Deployment - Once the software is certified, and no bugs or errors are stated, then

it is deployed. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.

1. Maintenance - Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

## SYSTEM IMPLEMENTATION PLAN

The System Implementation plan table, shows the overall schedule of tasks compilation and time duration required for each task.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Name/Title** | **Time Span** |  |
| 1 | Preliminary Survey | 1 Week |  |
| 2 | Introduction and Problem Statement | 1 Week |  |
| 3 | Literature Survey | 2 Weeks |  |
| 4 | Project Statement | 1 Week |  |
| 5 | Software Requirement And  Specification | 2 Week |  |
| 6 | System Design | 2 Week |  |
| 7 | Partial Report Submission | 2 Weeks |  |
| 8 | Architecture Design | 2 Weeks |  |
| 9 | Implementation | 2 Weeks |  |
| 10 | Deployement | 1 Week |  |
| 11 | Testing | 1 Week |  |
| 12 | Paper Publish | 1 Week |  |
| 13 | Report Submission | 1 Week |  |

# CHAPTER 4 PROJECT PLAN

## RECONCILED ESTIMATES

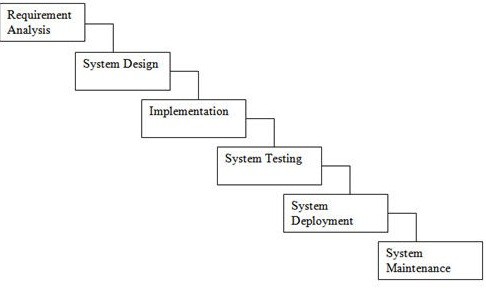


Figure 4.1

1. **Requirement gathering and analysis:** In this step of waterfall we identify what are various requirements are need for our project such are software and hardware required, database, and interfaces.
2. **System Design:** In this system design phase we design the system which is easily understood for end user i.e. user friendly. We design some UML diagrams and data flow diagram to understand the system flow and system module and sequence of execution.
3. **Implementation:** In implementation phase of our project we have implemented various module required of successfully getting expected outcome at the different module levels. With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
4. **Testing:** The different test cases are performed to test whether the project module are giving expected outcome in assumed time. All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
5. **Deployment of System:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
6. **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards like a waterfall through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name “Waterfall Model”. In this model phases do not overlap.

## Project Resource

Well configured Laptop, Anaconda Navigator, Spyder IDE 2 GHZ CPU speed, 2 GB RAM, Internet connection

## RISK MANAGEMENT

1. **In appropriate dataset -**To overcome this risk we are trying to use well organized and complete dataset.
2. **Security-** To overcome and improving security we use multilevel security like access permissions of user.

## Risk Identification

1. Have top software and customer managers formally committed to support the project?

Answer-Not applicable.

1. Are end-users enthusiastically committed to the project and the system/product to be built?

Answer-Not known at this time.

1. Are requirements fully understood by the software engineering team and its customers?

Answer-Yes

1. Have customers been involved fully in the definition of requirements? Answer-Not applicable
2. Do end-users have realistic expectations? Answer-Not applicable
3. Does the software engineering team have the right mix of skills? Answer-yes
4. Are project requirements stable? Answer-Not applicable
5. Is the number of people on the project team adequate to do the job? Answer-Not applicable
6. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

Answer-Not applicable

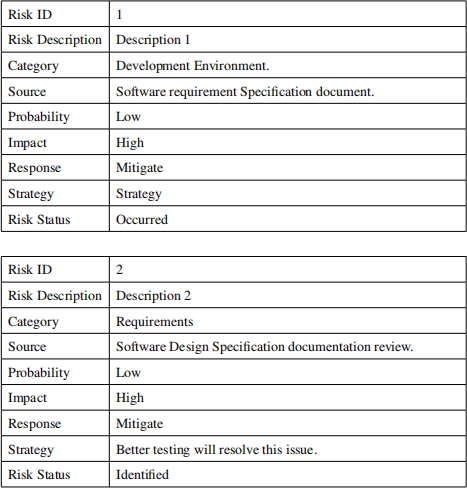
## Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

## Risk Mitigation Risk Monitoring and Risk Management

## 

## Figure 4.2 Risk Mitigation Risk Monitoring and Risk Management



## PROJECT SCHEDULE

## Project Task Set

Major Tasks in the Project stages are:

* + - * Task 1: correctness
      * Task 2: availability
      * Task 3: integrity

## Task Network

Figure 4.3: Task Network

## Timeline Chart

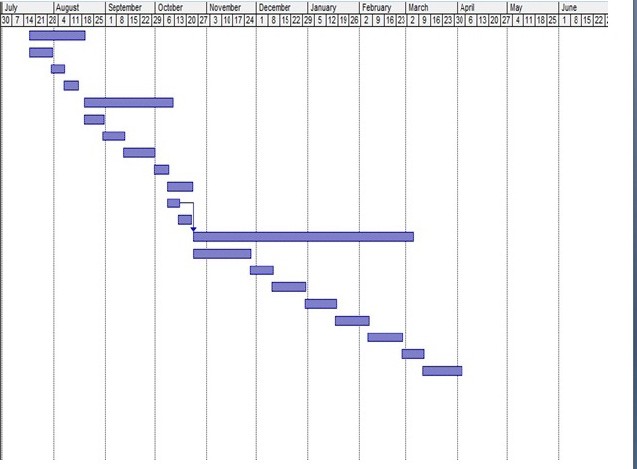


Figure 4.4: Timeline Chart

## TEAM ORGANIZATION

Team consists of 4 members and proper planning mechanism are used and roles of each member are defined.

## Team Structure

The team structure for the project is identified. There are total 4 members in our team and roles are defined. All members are contributing in all the phases of project.

## Management Reporting And Communication

Well planning mechanisms are used for progress reporting and inter/intra team communication are identified as per requirements of the project.

# CHAPTER 5 SYSTEM DESIGN

## SYSTEM ARCHITECTURE

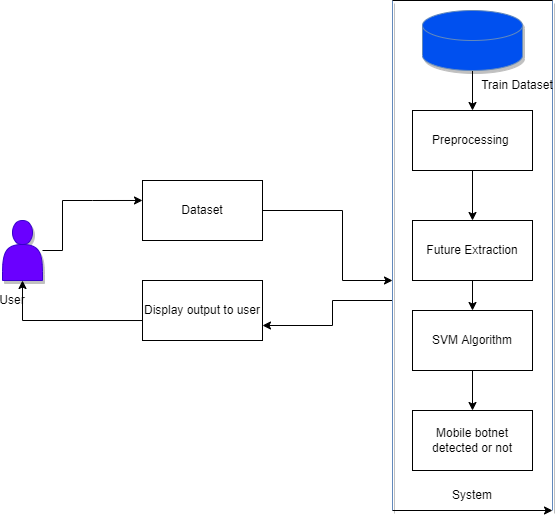


Figure 5.1: System Architecture

SVM can be explained through an example as follows, suppose we encounter a strange weird cat that also has some attributes or characteristics of a dog, so now if we require a model that would precisely identify whether the given animal is a cat or a dog, such a model is perfectly possible by an SVM algorithm, Firstly we shall train our model with numerous images of cats and dogs, for it to learn about their different and distinguishing features. After that, we will test it with a strange creature, so now as explained earlier the SVM will create a boundary of decision between these two different pieces of data i.e. cat and dog, and chooses the extreme scenarios (support vectors), evidently, it will see the extreme instance of cat and a dog then basing its decision on support vectors, the classification will be done as a cat or a dog. The dimensions of a hyperplane are contingent on the characteristics in the dataset, this says if there exist two features, the hyperplane should be a straight line. And if there are 3 features, then it comes out to be a 2-dimensional plane. These support vectors can classify new data. When the data is not linearly separable, the kernel function should be used to map the data into the Vapnik Chervonenkis dimensional cavity

## DATA FLOW DIAGRAMS

## Data Flow Diagram

In Data Flow Diagram, we Show that flow of data in our system in DFD0 we show that base DFD in which rectangle present input as well as output and circle show our system, In DFD1 we show actual input and actual output of system input of our system is text or image and output is rumor detected likewise in DFD 2 we present operation of user as well as admin.



Figure 5.2: Data Flow0 diagram

Figure 5.3: Data Flow 1diagram

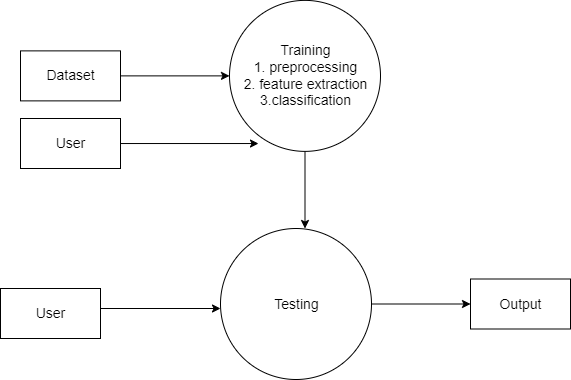
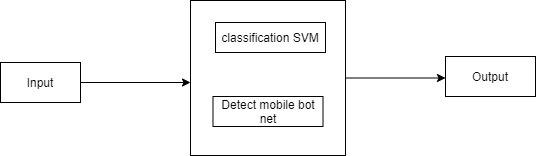


Figure 5.4: Data Flow2 diagram

**Usecases Diagram**

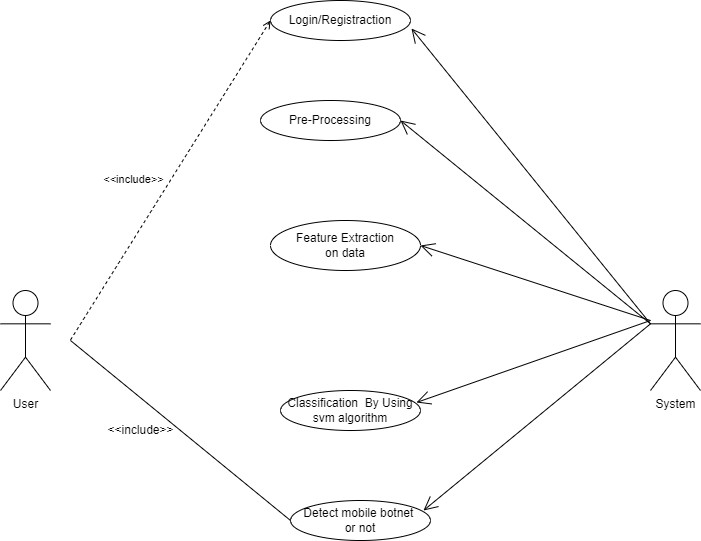


Figure 5.5: Usecase Diagram

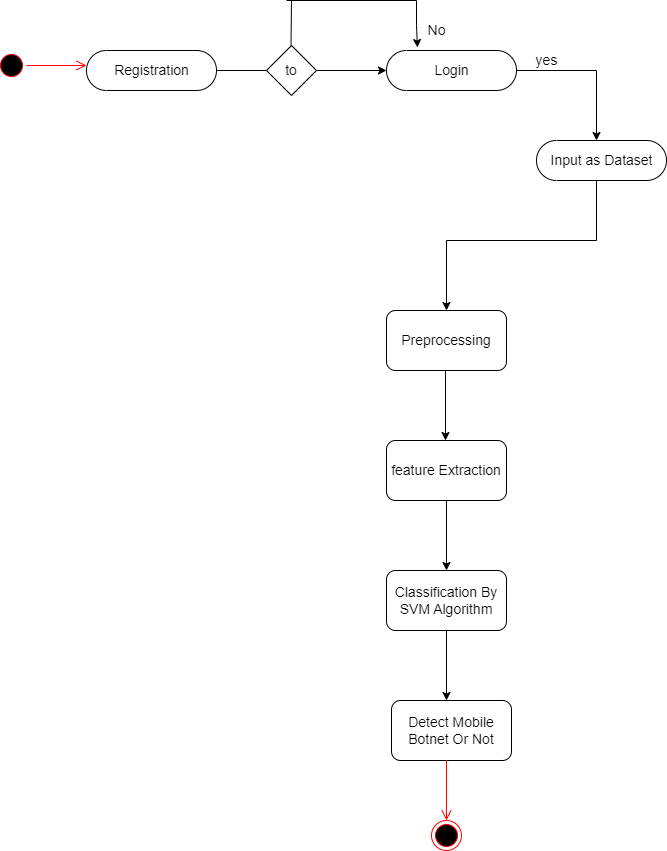


Figure 5.6: Activity Diagram

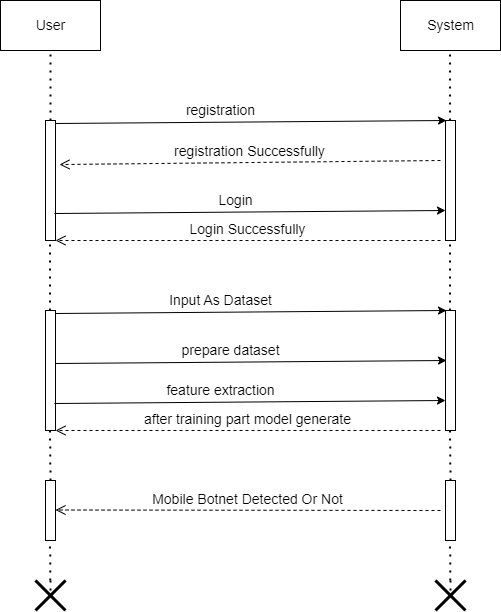


Figure 5.7: Sequence Diagram

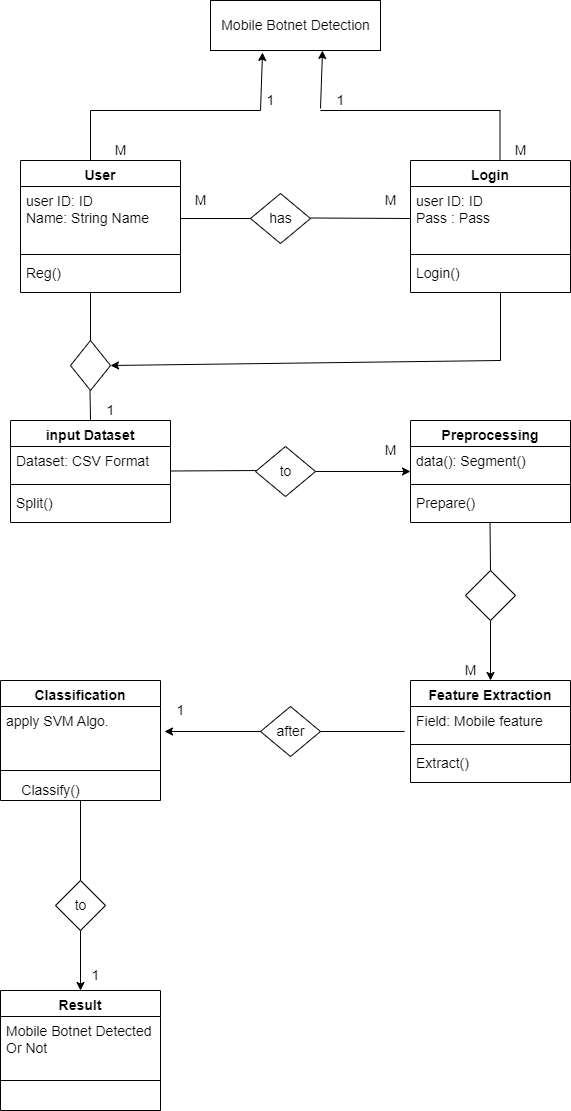


Figure 5.8: Class Diagram

# CHAPTER 6 SOFTWARE TESTING

## PURPOSE

Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. Software testing can also be stated as the process of validating and verifying that a software program or application or product:

* + 1. Meets the business and technical requirements that guided its design and development;
    2. Works as expected; and
    3. Can be implemented with the same characteristics

## SCOPE OF TESTING

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development method- ology adopted. Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

## SOFTWARE TO BE TESTED

1. **Edraw Max:**

It enables students, teachers and business professional store liable create and publish various kinds of diagram store present any ideas. With this application user can easily create professional- looking flow charts, organizational charts, network

diagrams, business presentations, building plans, mind maps, science illustration, fashion designs, UML diagrams and much more.

## Star UML:

Star UML is a fully fledged, open source, UML modeling tool that’s supports the ability to create software designs, from basic concepts, through to the coded solution. The user should be aware that this tool is more complex than a simple UML diagram editing tool, in that, through the use of the model Drive Architecture (MDA) standard, the tool supports complex modeling which is realizable in code.

## TEST PLAN

To test this application, we are going with proper sequencing of testing like unit, integration, validation, GUI, Low level and High level test cases, major scenarios likewise. We will go with the GUI testing first and then integration testing. After integration testing performs the high level test cases and major scenarios which can affect the working on the application. We will perform the testing on the data trans- mitted using the various inputs and outputs and validate the results. It also intends to cover any deviations that the project might take from the initially agreed Test Strategy in terms of scope, testing methodology, tools, etc. This test plan covers details of testing activities for this project and scope.

## TYPE OF TESTING

## Unit Testing

It is the testing of individual software units of the application .it is done after the complexion of an individual unit before integration. Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of

business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## Regression Testing

Regression testing is a software testing practice that ensures an application still functions as expected after any code changes, updates, or improvements. Regression testing is responsible for the overall stability and functionality of the existing features.

## Smoke Testing

Smoke Testing comes into the picture at the time of receiving build software from the development team. The purpose of smoke testing is to determine whether the build software is testable or not. It is done at the time of” building software.” This process is also known as” Day 0”.

It is a time-saving process. It reduces testing time because testing is done only when the key features of the application are not working or if the key bugs are not fixed. The focus of Smoke Testing is on the workflow of the core and primary functions of the application.

## System Testing

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

# CHAPTER 7 TECHNICAL SPECIFICATION

## ADVANTAGES

1. Mobile botnets take advantage of unpatched exploits to provide hackers with root permissions over the compromised mobile device, enabling hackers to send email or text messages, make phone calls, access contacts and photos, and more
2. Easy To Use

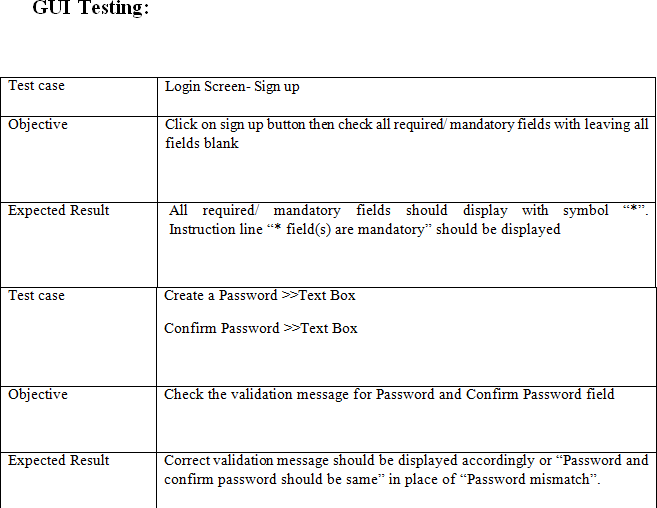
## DISADVANTAGES:

1. Botnets are a big piece of the pie when it comes to malware infection and distribution across the internet, which leads to everything from theft of data and credentials to website destruction and extortion.

## APPLICATION

1. A mobile botnet is a number of Internet-connected devices, each of which is running one or more bots. Botnets can be used to perform Distributed Denial of Service (DDoS) attacks, steal data, send spam, and allow the attacker to access the device and its connection.

# CHAPTER 8 TEST CASES



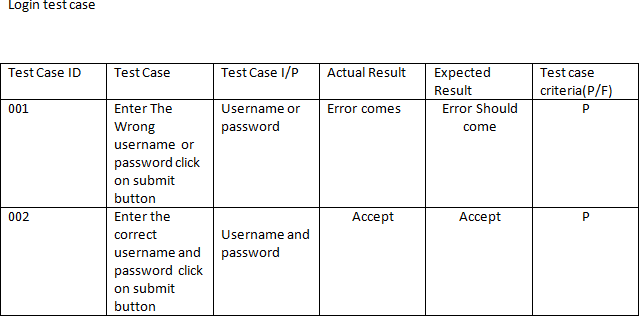
Figure 8.1: GUI Testing

Figure 8.2: login test Cases

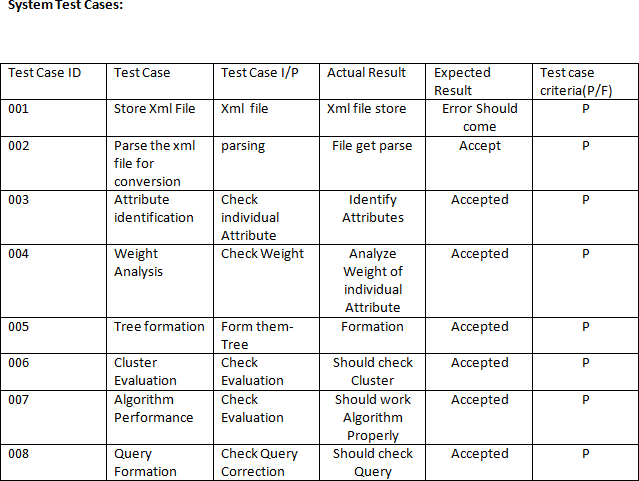
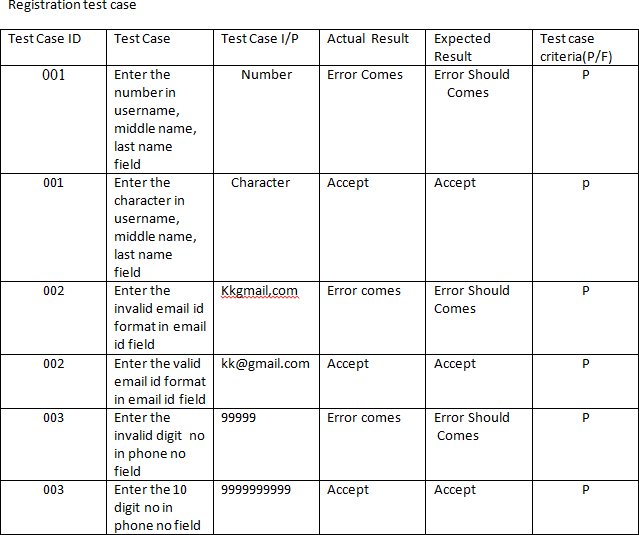
Figure 8.3: Registration Test cases

Figure 8.4: System Test Cases

# CHAPTER 9 RESULT

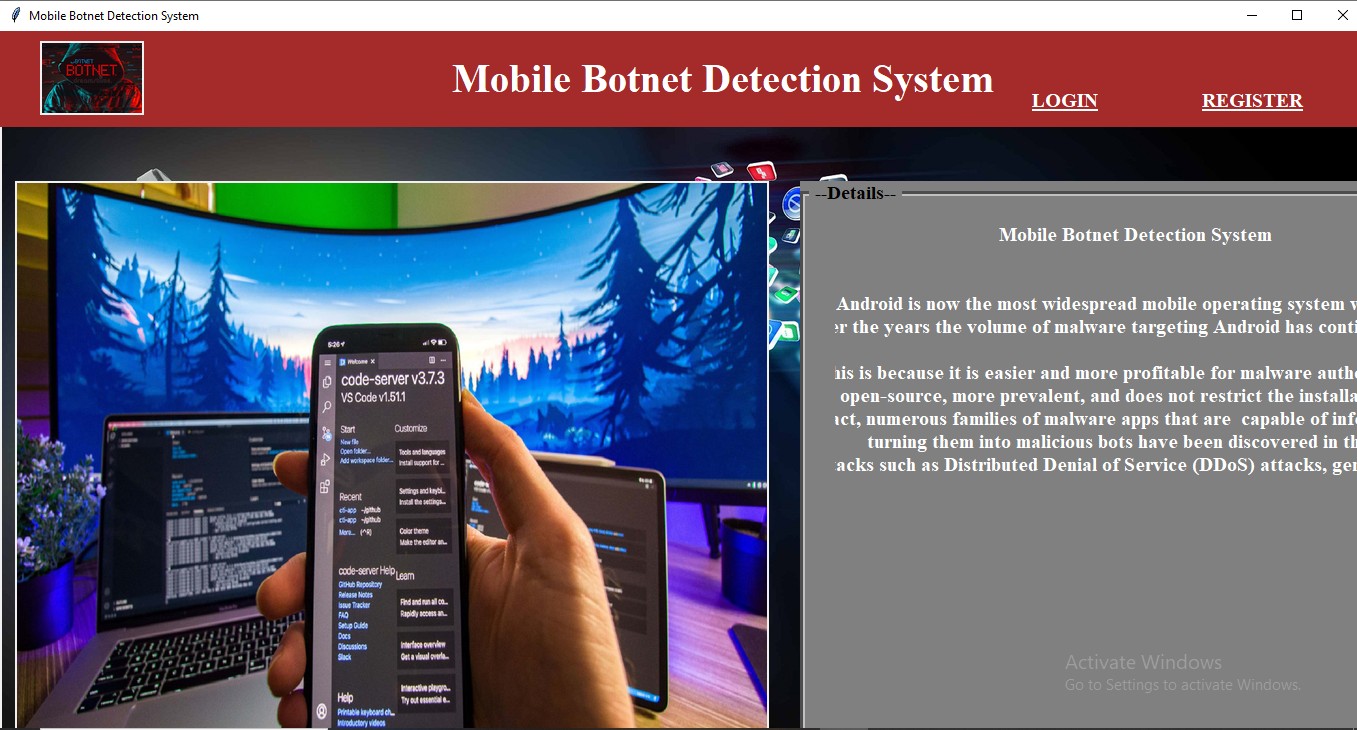


Figure 9.1: main page

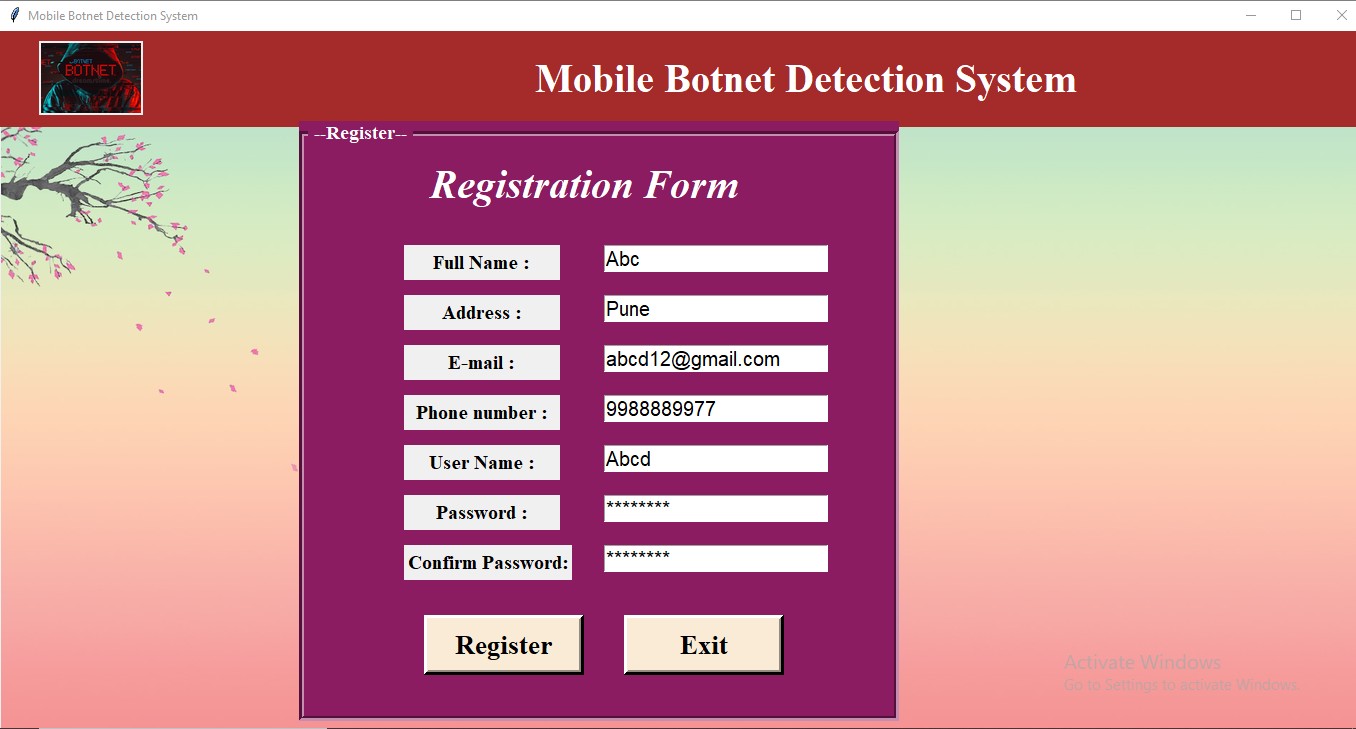


Figure 9.2: registration page

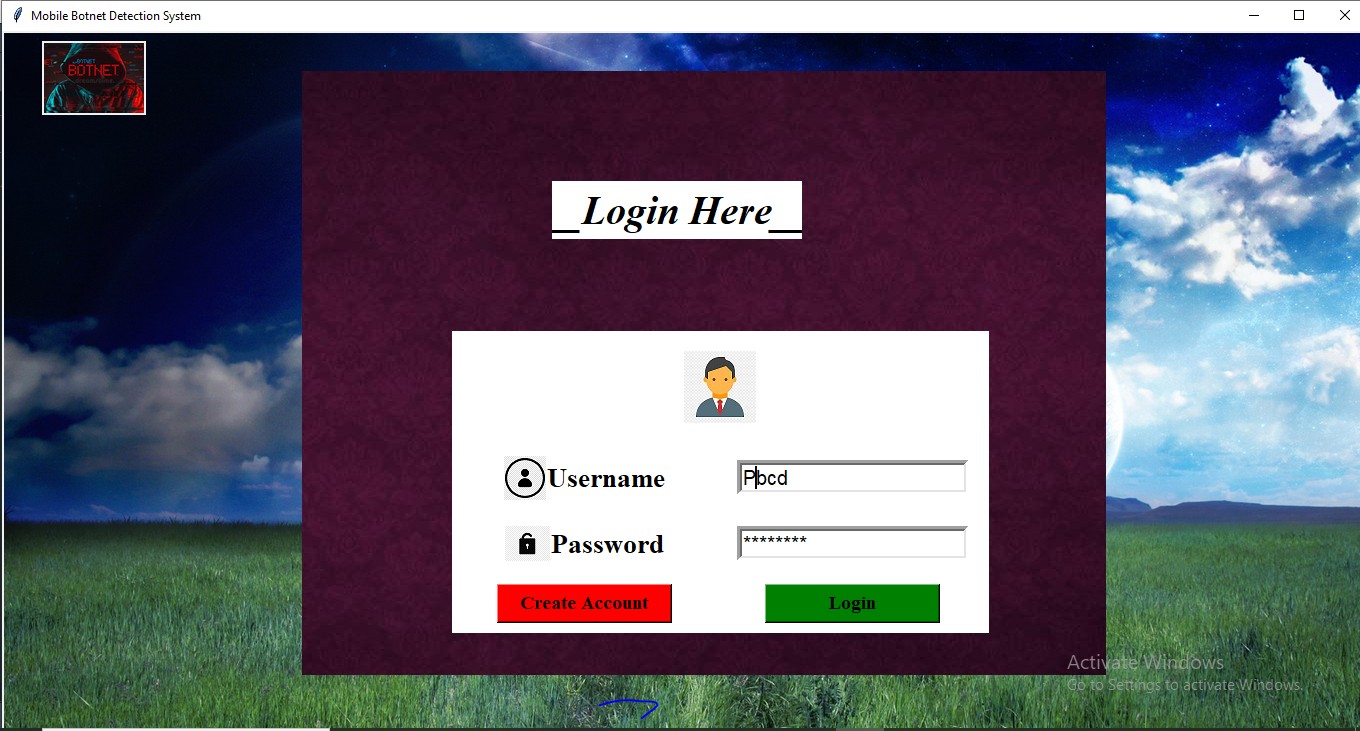


Figure 9.3: login page



Figure 9.4: output1

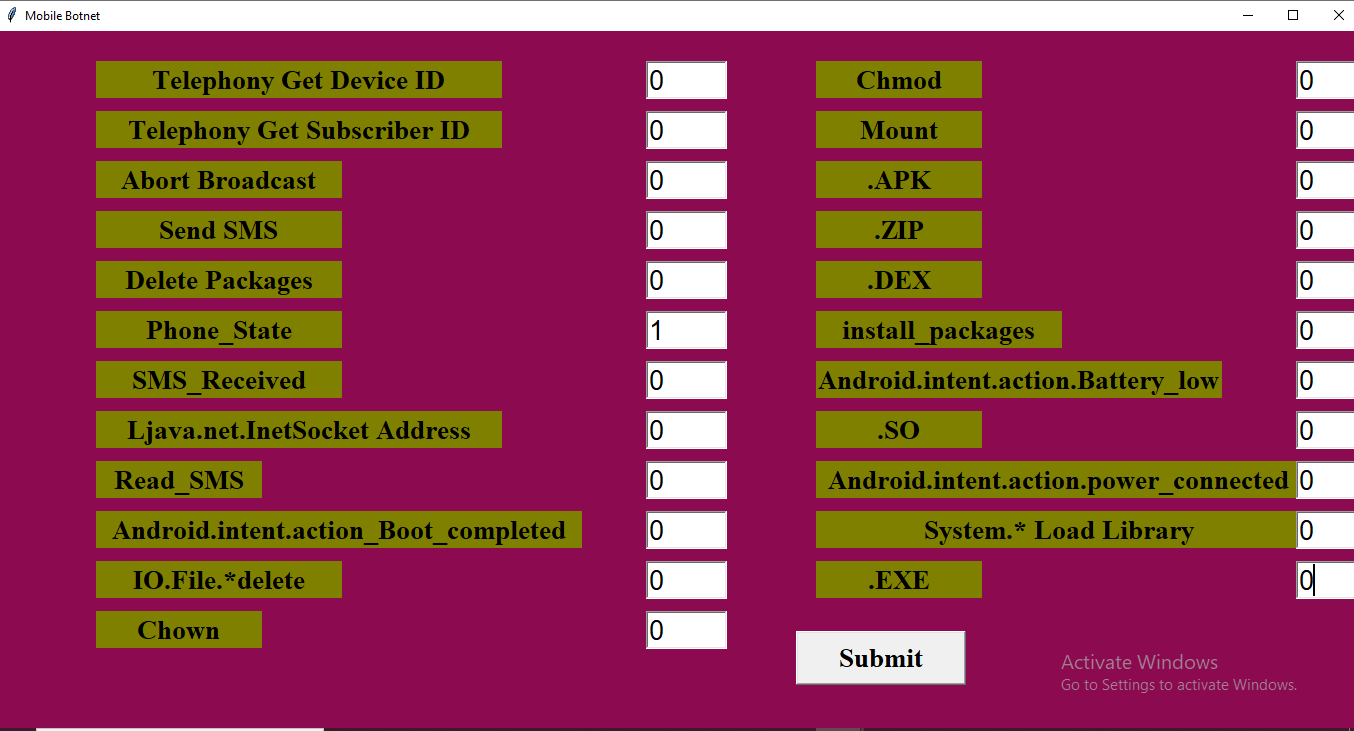


Figure 9.5: output2

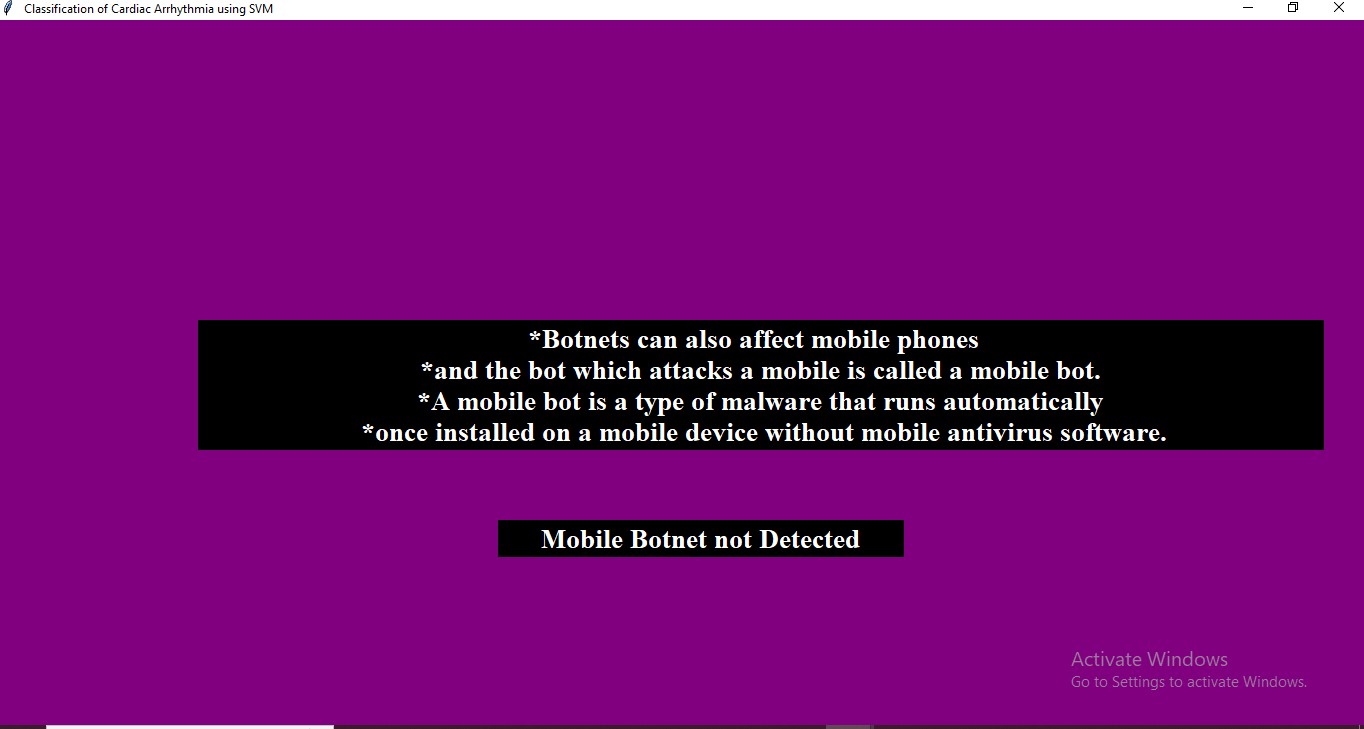


Figure 9.6: output3

# CHAPTER 10 CONCLUSIONS

Botnets are a Dangerous evolution in the malware world. They are being used to damage systems, steal information and Comprise Systems. They are hard to detect and eliminate. So Our System Is Useful to detect Mobile Botnet.

## FUTURE WORK

malicious programs are discovered in the future, we may add more data sets to make it even more precise and easier to use. With these features, it may be utilized in higher-level platforms such as Play store and App store, even if it cannot hold or achieve that level. We can also use it in local company systems to ensure that all apps and programs are malware- free.

# CHAPTER 11 REFERENCES

* + 1. S. Y. Yerima and S. Khan “Longitudinal Performance Analysis of Machine Learning based Android Malware Detectors” 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE.
    2. Kadir, A.F.A., Stakhanova, N., Ghorbani, A.A., 2015. Android botnets: What urls are telling us, in: International Conference on Network and System Security, Springer. pp. 78–91.
    3. ISCX Android botnet dataset. Available from [https://www](http://www.unb.ca/cic/).unb[.ca/cic/](http://www.unb.ca/cic/) datasets/android- botnet.html. [Accessed 03/03/2020]
    4. M. Eslahi, R. Salleh, and N. B. Anuar,” Bots and botnets: An overview of characteristics, detection and challenges,” in Proceedings of the IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2012, pp. 349-354.
    5. J. Dae-il, C. Kang-yu, K. Minsoo, J. Hyun-chul, and N. Bong-Nam,” Evasion technique and detection of malicious botnet,” in Proceedings of the Internet Technology and Secured Transactions (ICITST), 2010 International Conference for, 2010, pp. 1-5.
    6. Z. Yajin and J. Xuxian,” Dissecting Android Malware: Characterization and Evolution,” in Proceedings of the Symposium on Security and Privacy (SP), 2012, pp. 95-109.
    7. M. Eslahi, M. Rohmad, H. Nilsaz, M. Var Naseri, N. Tahir, and H. Hashim,” Periodicity classification of HTTP traffic to detect HTTP Botnets,” in Proceedings of the Computer Applications Industrial Electronics (ISCAIE), 2015

IEEE Symposium on, 2015, pp. 119-123.

* + 1. T. Strazzere and T. Wyatt,” Geinimi trojan technical teardown,” Lookout Mo- bile Security, 2011.
    2. G. Gu, R. Perdisci, J. Zhang, and W. Lee,” BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection,”in Proceedings of the 17th conference on Security symposium, San Jose, CA, 2008, pp. 139-154.
    3. M. Eslahi, R. Salleh, and N. B. Anuar,” MoBots: A new generation of botnets on mobile devices and networks,” in Proceedings of the IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), 2012, pp. 262- 266.