A Project Report on

Android Malware Detection using Machine Learning

Submitted in partial fulfillment of the requirements for the award
of the degree of

Bachelor of Engineering

in

Information Technology

by

Rishab Agrawal(17204006)
Vishal Shah(17204003)
Sonam Chavan(16104067)

Under the Guidance of

Mr. Ganesh Gourshete
Ms. Nahid Shaikh

Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2019-2020

Approval Sheet

This Project Report entitled ***"Android Malware Detection using Machine Learning"*** Submitted by ***"Rishab Agrawal"(17204006),"Vishal Shah"(17204003),"Sonam Chavan"(16104067)***is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

Ms.Nahid Shaikh
Co-Guide

Mr.Ganesh Gourshete
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Android Malware Detetction using Machine Learning"*** submitted by ***"Rishab Agrawal" (17204006),"Vishal Shah" (17204003),"Sonam Chavan" (16104067)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology.***,to the University of Mumbai,is a bonafide work carried out during academic year 2019-2020.

Ms. Nahid Shaikh
Co-Guide

Mr. Ganesh Gourshete
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Rishab Agrawal 17204006)

_____

(Vishal Shah 17204003)

_____

(Sonam Chavan 16104067)

Date:

Abstract

Malware is one of the major issues regarding the operating system or in the software world. The android system is also going through the same problems. We have seen other Signature based malware detection techniques were used to detect malware. But the techniques were not able to detect unknown malware. Despite numerous detection and analysis techniques are there, the detection accuracy of new malware is still a crucial issue. In this paper, we study and highlight the existing detection and analysis methods used for the android malicious code. Along with studying, we propose Machine learning algorithms that will be used to analyze such malware and also we will be doing semantic analysis. We will be having a data set of permissions for malicious applications. Which will be compared with the permissions extracted from the application which we want to analyze. In the end, the user will be able to see how much malicious permission is there in the application and also we analyze the application through comments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Malware is nothing but the short name for malicious software, in general, referred to many forms of hostile or intrusion creating software, spyware, Trojan horses, backdoor, and rootkits. The main aim of malware is to damage, steal, disrupt or do some bad actions. Malware is powerful enough to infect any kind of computing machine running application, and the prevention of malware is being well studied for personal computers. A Smartphone device the detection techniques used is lagging far behind as compared to the fast growth of the mobile population is being Some recent survey has shown that there are about 2.1 million android applications are there in the market. Due to increase in usage of the android system has led to more rollout of android malware. This malware is spreading in the market by the third parties developing applications. The Google android market also doesn't promise to guarantee that all the applications listed are threat free. There are also such reports about Trojans applications that if downloaded, their malicious code is also installed and cannot be easily detected by Google's technologies during publication in the Google android market. The android threats include banking Trojans, spyware, bots, root exploits, SMS fraud, phishing fake installer.
Also semantic analysis describes the process of understanding natural language–the way that humans communicate–based on meaning and context.The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.So with the help of semantic analysis we are able to get the comments being ordered as positive,negative and neutral type of comment and from that we are able to get the semantic analysis of that particular application.

1.1 Objective

The objective behind developing this project is:

- To have an malware detection interface
- To provide user use our interface.
- To make service available whenever they required.
- Instead of relying what the permissions the user have granted our project will be able to list all the permissions over it.
- Also the permissions will be classify later and according to it the malware will be detected from the dataset.
- Also semantic analysis will be there which judge the app from the comments the user has been posted.

1.2 Problem Statement

- The Existing system was working on the different Apk files where they were not getting the actual and the perfect output of the malware files as required.The existing users has to go on the terminal and they were not getting the accuracy as the datasets were not been trained.In this system there is more accuracy even the datasets to be taken is of bulk so the user can get the actual output needed for the particular malware of the particular Apk file.Also we are developing an User Interface where the user can interact and also he can be able to get the permissions which the user has provided to the particular application.And there would be semantic analysis which will based on the comments the particular user has posted and it will give the results on it.

1.3 Malware insertion

- Repacking:- It is among the common techniques for malware developers to install malicious applications on a android platform. Repackaging approach for popular applications and misuse them as a malware. The developer downloads such types of application and recode them and add their own malicious code and upload that application to the official android app store or on the different markets.
- Updating:- This technique is much more difficult for detecting the malware. The malware developer may still use repackaging but instead of encoding the inflict code to the application,the developer may include a update component that will able to download malicious code at the run time.
- Downloading:- This is the most traditional attacking technique. The malware developer need to attract the user to download the interesting and attractive applications.

1.4 User Interface

- Admin side to manage the whole website for granting and revoking privileges to the other entities of the website.Also adding of new application,comments,etc are been taken care by the admin side.
- User Portal will access the website and would be able to watch which of the application is malware affected or not.

1.5 Machine learning classifier

- Machine learning (ML) is the study of computer algorithms that improve automatically through experience.Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. So with the help of classifiers we are able to detect the malware. We have used some of the permissions which are been used as an malware permissions and fro that we have classified through our data in which we used supervised learning classifier.

1.6 Semantic analysis

- Semantic analysis describes the process of understanding natural language–the way that humans communicate–based on meaning and context.
- The semantic analysis of natural language content starts by reading all of the words in content to capture the real meaning of any text.
- It identifies the text elements and assigns them to their logical and grammatical role. It analyzes context in the surrounding text and it analyzes the text structure to accurately disambiguate the proper meaning of words that have more than one definition.
- So in our semantic analysis it analyzes the comments of the app from the users and then it gives weather the comment is positive,negative or neutral.

1.7 Storage

- Some minor fields are been stored like the name of the application,type of application,comments,etc. Also in the database the reviews are been stored which is an semantic analysis part in our project.And from the database we used to fetch that all the data in to our classification and analysis in the result.

1.8 Technology Stack

- Hardware And Software Requirements.

Hardware Requirements:

Processor : Intel Dual Core/Above.
Monitor : LCD Colour.
Mouse : Optical Mouse.
RAM : 2 GB

Software Requirements:

Operating System : Windows 07 and above.
Coding Language : JAVA
Java Version : JDK1.6 and Above
Database : MySQL
Tool : Eclipse
Front End : Java
Back End : MySQL

Chapter 2

Literature Review

[1]Darus, Fauzi Mohd, Salleh Noor Azurati Ahmad, and Aswami Fadillah Mohd Ariffin. "Android Malware Detection Using Machine Learning on Image Patterns." 2018 Cyber Resilience Conference (CRC). IEEE, 2018.-In this paper we have seen that they had 300 malware and 300 benign APK files, and they man-aged to generate only 183 malware and 300 benign gray-scale images. The other 117 malware samples could not be generated into images because the APK files are either corrupted or they did not contain classes.dex file.And their accuracy was less in all the algorithms they have used.Also there are two types of detection techniques that are normally used by malware analysts, static and dynamic detection. Static detection is based on specific strings from the disassembled code without executing the binary file. This analysis can quickly capture the syntax but it is easily disturbed by code obfuscation and encryption technology. The second type of detection is dynamic detection. It analyses the malware behaviour such as network activities, system calls, and file operations by executing the malware. This technique can detects newly created malware however, it requires more execution time.

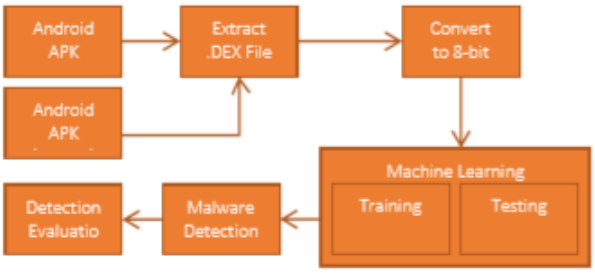


Figure 2.1: Android Malware image detection(Darus-2018)

Figure 3.1 explains that they have an an Android Apk file from there they have extracted the .DEX file from the Apk file and then convert the .DEX file in to an 8-bit file.The 8-bit file is been trained with the help of machine learning algorithms and then testing is been done in the machine learning algorithms. By training and testing the malware detection is been done on the basis of the results given by the algorithms and from there, the detection of the malware is been done and the evaluation is based on the datasets of a particular malware files.

[2]Varma, P. Ravi Kiran, Kotari Prudvi Raj, and KV Subba Raju. "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms." 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2017.-In this paper different machine learning algorithms are used such as navies Bayes, j48, Randomforest, Multi class classifier and multi-layer perceptron to detect android malware and evaluate the performance of each algorithm. Here they implemented a framework for classifying android applications with the help of machine learning techniques to check whether it is malware or normal application. To access this model have to extract several features and permission from many downloaded applications from the android market. For validating, their system they collected 3258 samples of android apps and those have to be extracted for each and every application, extract their features and have to train the models going to be evaluated with the help of classification accuracy and time taken for the model.In order to use machine learning algorithms in classification of android malware, first the permission data set of all the apps are to be extracted. Database is formed by placing a 1 if the permission is present in the Androidmanifest.And from this they have used the algorithm and got the results.

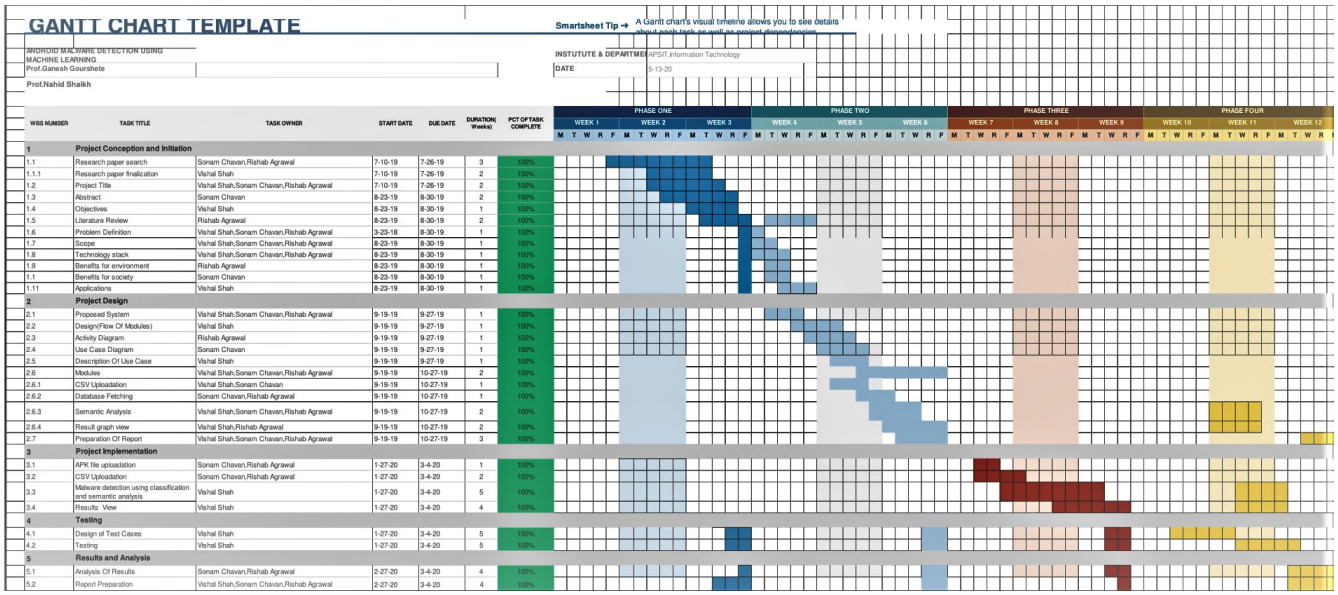
[3]Chang, Wei-Ling, Hung-Min Sun, and Wei Wu. "An Android Behavior-Based Malware Detection Method using Machine Learning." 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). IEEE, 2016.-In this they have proposed a Robotium program in an Android sandbox which can trigger Android application automatically and monitor behavior. A Robotium program is a UI-identication automatic trigger program which can click mobile applications in meaningful order.More importantly, they do large scale experiments. They build a decision model using behaviour we collected and RandomForest algorithm. It can determine whether unknown application is a malware and show its condence value. Their evaluation indicators show permission and dynamic behavior we collected such as network activity, read/write, phone call, sent SMS, in-formation leak are suitable feature data for classification malware. The accuracy of their model is 97 and the FPR is less than 4. They also store classification result and condence value of unknown APK in their database.

Figure 2.2: System Framework(Chang-2016)

Figure 3.2 describes their System Framework which defines the whole process of their as they have an APK file from there they have done the preprocessing of that particular APK. Also there is an Data Monitor which monitors the whole process and from there the Decision model modifies the whole data and the result is been stored in the database.

[4]Koli, J. D. "RanDroid: Android malware detection using random machine learning classifiers." 2018 Technologies for Smart-City Energy Security and Power (ICSESP). IEEE, 2018-In this, Android malware detection system is proposed which uses permission,APIs,and presence of others key apps information such as,dynamic code,reaction code,native code, cryptographic code, database etc. as features to train and build classification model by using various techniques which automatically distinguish malicious Android apps (malware) from legitimate ones.In this they used risky permission and vulnerable API calls as feature to train SVM classifier to detect malware in android application. It uses set of 400 android app to train a SVM model. Then it uses set of 300 apps as testing set to evaluate the accuracy of classification model. The android malware detection system was proposed which uses permission, APIs, and presence of others key apps information such as, dynamic code, reflection code, native code, cryptographic code, database etc. as features to train and build classification model by using various machine learning techniques which can automatically distinguish malicious Android apps (malware) from legitimate ones

Gantt Chart



GANTT CHART TEMPLATE

Smartsheet Tip → A Gantt chart's visual timeline allows you to see details about each task as well as project dependencies.

ANDROID MALWARE DETECTION USING MACHINE LEARNING
Prof.Ganesh Gourshete
Prof.Nahid Shaikh

INSTUTUTE & DEPARTMEN APSIT,Information Technology
DATE 5-13-20

PHASE ONE: WEEK 1, WEEK 2, WEEK 3 | PHASE TWO: WEEK 4, WEEK 5, WEEK 6 | PHASE THREE: WEEK 7, WEEK 8, WEEK 9 | PHASE FOUR: WEEK 10, WEEK 11, WEEK 12

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION(Weeks)	PCT OF TASK COMPLETE
1	**Project Conception and Initiation**					
1.1	Research paper search	Sonam Chavan,Rishab Agrawal	7-10-19	7-26-19	3	100%
1.1.1	Research paper finalization	Vishal Shah	7-10-19	7-26-19	2	100%
1.2	Project Title	Vishal Shah,Sonam Chavan,Rishab Agrawal	7-10-19	7-26-19	2	100%
1.3	Abstract	Sonam Chavan	8-23-19	8-30-19	2	100%
1.4	Objectives	Vishal Shah	8-23-19	8-30-19	1	100%
1.5	Literature Review	Rishab Agrawal	8-23-19	8-30-19	2	100%
1.6	Problem Definition	Vishal Shah,Sonam Chavan,Rishab Agrawal	3-23-18	8-30-19	1	100%
1.7	Scope	Vishal Shah,Sonam Chavan,Rishab Agrawal	8-23-19	8-30-19	1	100%
1.8	Technology stack	Vishal Shah,Sonam Chavan,Rishab Agrawal	8-23-19	8-30-19	1	100%
1.9	Benefits for environment	Rishab Agrawal	8-23-19	8-30-19	1	100%
1.1	Benefits for society	Sonam Chavan	8-23-19	8-30-19	1	100%
1.11	Applications	Vishal Shah	8-23-19	8-30-19	1	100%
2	**Project Design**					
2.1	Proposed System	Vishal Shah,Sonam Chavan,Rishab Agrawal	9-19-19	9-27-19	1	100%
2.2	Design(Flow Of Modules)	Vishal Shah	9-19-19	9-27-19	1	100%
2.3	Activity Diagram	Rishab Agrawal	9-19-19	9-27-19	1	100%
2.4	Use Case Diagram	Sonam Chavan	9-19-19	9-27-19	1	100%
2.5	Description Of Use Case	Vishal Shah	9-19-19	9-27-19	1	100%
2.6	Modules	Vishal Shah,Sonam Chavan,Rishab Agrawal	9-19-19	10-27-19	2	100%
2.6.1	CSV Uploadation	Vishal Shah,Sonam Chavan	9-19-19	10-27-19	1	100%
2.6.2	Database Fetching	Sonam Chavan,Rishab Agrawal	9-19-19	10-27-19	1	100%
2.6.3	Semantic Analysis	Vishal Shah,Sonam Chavan,Rishab Agrawal	9-19-19	10-27-19	2	100%
2.6.4	Result graph view	Vishal Shah,Rishab Agrawal	9-19-19	10-27-19	2	100%
2.7	Preparation Of Report	Vishal Shah,Sonam Chavan,Rishab Agrawal	9-19-19	10-27-19	3	100%
3	**Project Implementation**					
3.1	APK file uploadation	Sonam Chavan,Rishab Agrawal	1-27-20	3-4-20	1	100%
3.2	CSV Uploadation	Sonam Chavan,Rishab Agrawal	1-27-20	3-4-20	2	100%
3.3	Malware detection using classification and semantic analysis	Vishal Shah	1-27-20	3-4-20	5	100%
3.4	Results View	Vishal Shah	1-27-20	3-4-20	4	100%
4	**Testing**					
4.1	Design of Test Cases	Vishal Shah	1-27-20	3-4-20	5	100%
4.2	Testing	Vishal Shah	1-27-20	3-4-20	5	100%
5	**Results and Analysis**					
5.1	Analysis Of Results	Sonam Chavan,Rishab Agrawal	2-27-20	3-4-20	4	100%
5.2	Report Preparation	Vishal Shah,Sonam Chavan,Rishab Agrawal	2-27-20	3-4-20	4	100%

Figure 2.3: Gantt Chart

Chapter 3

Project Design

1.System Architecture.

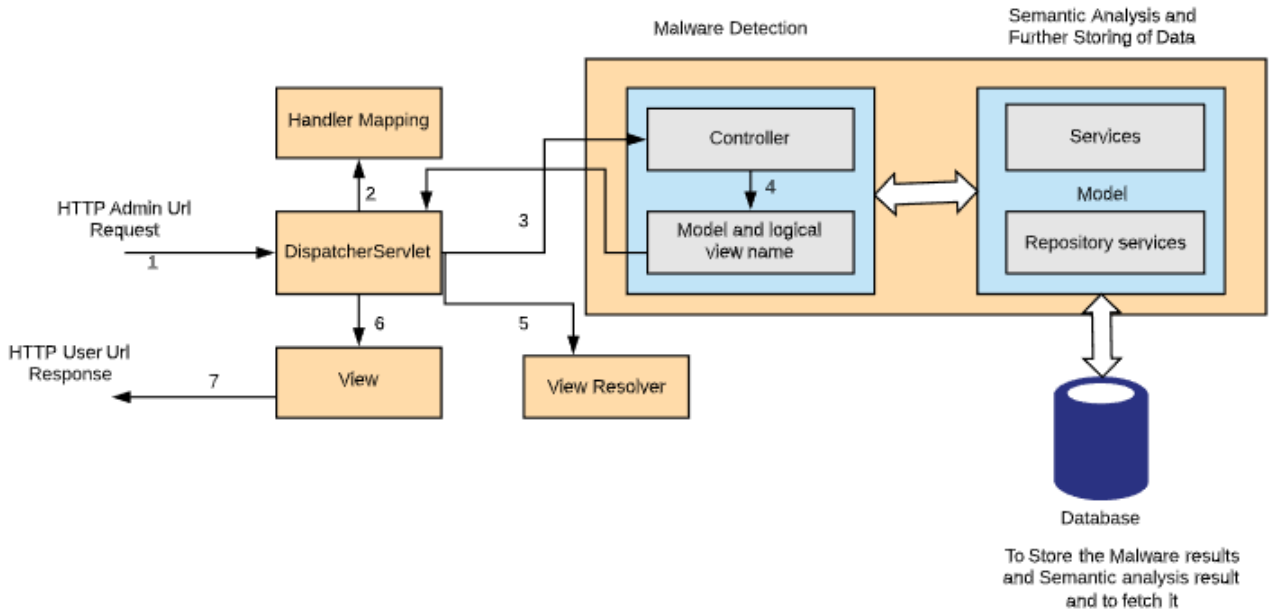


Figure 3.1: System Architecture for Proposed system

In the proposed system, we are doing the permission-based analysis and also the semantic analysis.

- The journey begins with the HTTP request (sometimes with data payload; for example, due to form submission) in a URL. It first stations at DispatcherServlet.
- The DispatcherServlet is a class defined in the org.springframework.web.servlet package. It is the central dispatcher, a Java Servlet Component for the Spring MVC framework. This front controller receives all incoming HTTP client requests and delegates responsibilities to other components for further processing of the request payload.

- The handler mapping decides where the request's next stop would be. It acts as a consultant to the central dispatcher (DispatcherServlet) for routing the request to the appropriate controller. The handler mapping parses the request URL to make decisions and the dispatcher then delegates the request to the controller.

- The controller's responsibility is to process the information payload received from the request. Typically, a controller is associated with one or more business service classes which, in turn, may have associated database services repository classes. The repository classes fetch database information according to the business service logic. It is the business service classes that contain the crux of processing. The controller class simply carries the information received from one or more service classes to the user. However, the response of the controller classes is still raw data referred to as the model and may not be user friendly (with indentation, bullets, tables, images, look-and-feel, and so forth).

- Therefore, the controller packages the model data along with model and view name back again to the central dispatcher, DispatcherServlet.

- The view layer can be designed using any third-party framework such as Node.js, Angular, JSP, and so on. The controller is decoupled from the view by passing the view name to the DispatcherServlet and is least interested in it. The DispatcherServlet simply carries the logical name and consults with the view resolver to map the logical view name with the actual implementation.

- Once the mapping between logical view name and the actual view implementation is made, the DispatcherServlet delegates the responsibility of rendering model data to the view implementation.

- The view implementation finally carries the response back to the client browser.

2.Use Case Diagram.

Figure 3.2: Use Case Diagram For Admin

In Use case diagram for admin, admin will be able to login, add Apk file, name of the Apk, give description for Apk, categories of an Apk, upload image for logo of an Apk. Admin can also add CSV files for comments and all will be stored in the database.

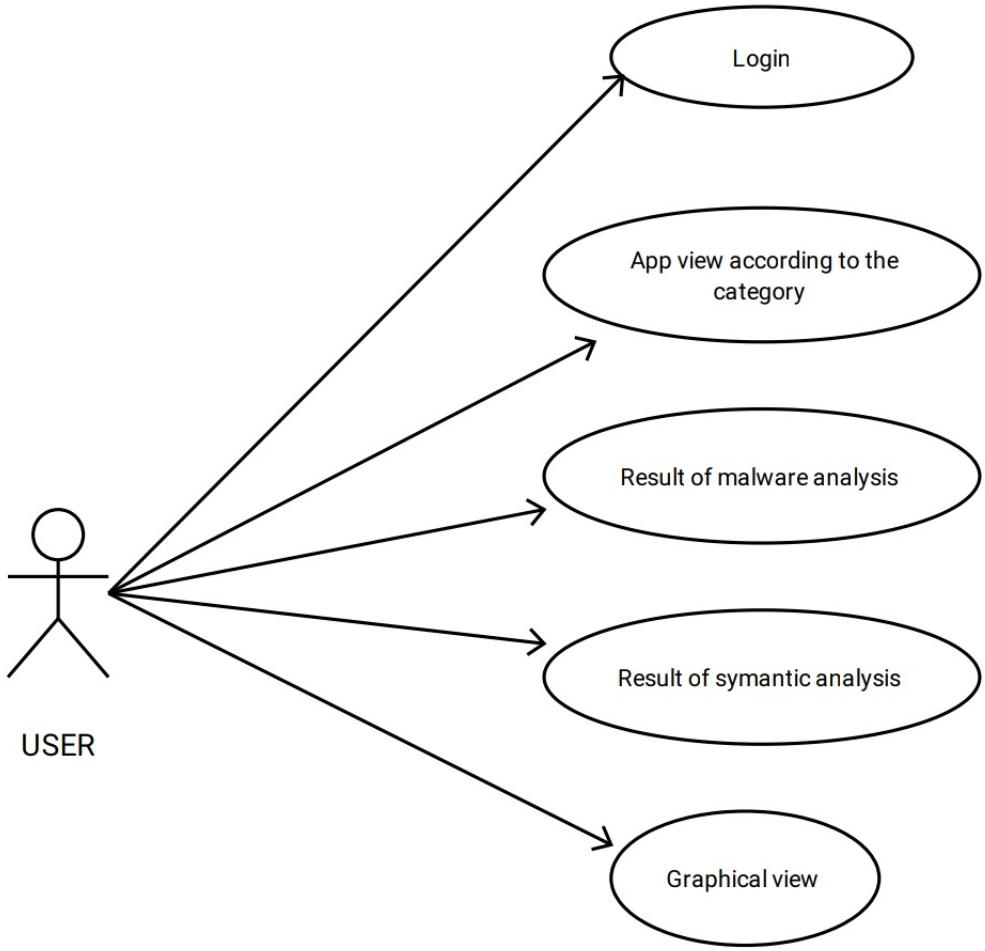


Figure 3.3: Use Case Diagram For User

In Use case diagram for user, user will be able to login and register, view Apk file, results of Malware analysis of the Apk, Result of Semantic analysis. Also there would be an Graphical view of the Apk file which shows the Malware percent of the Apk. Which will be fetched from the Database.

3.Class Diagram

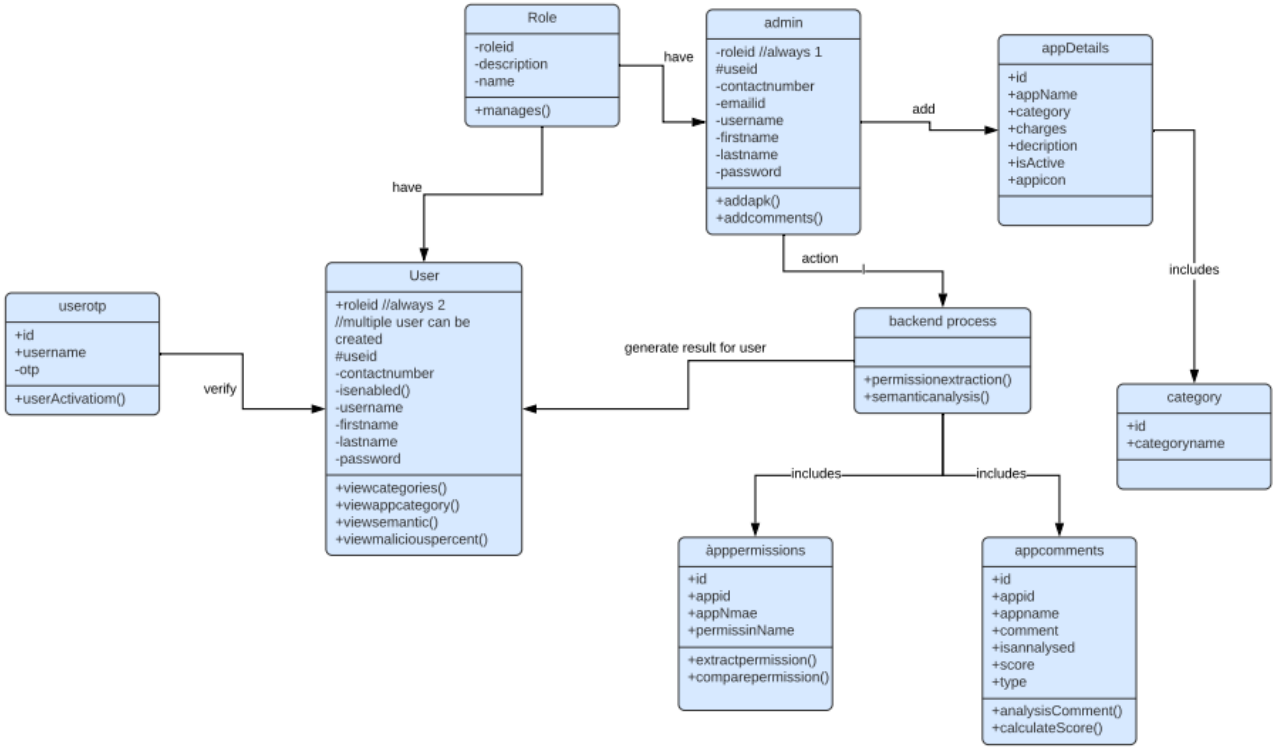


Figure 3.4: Class Diagram

A class diagram shows the relationship between the attributes and operations of all classes in the system.In our system the admin,user,app details, permissions all the classes are been connected to each other.

4.Activity Diagram

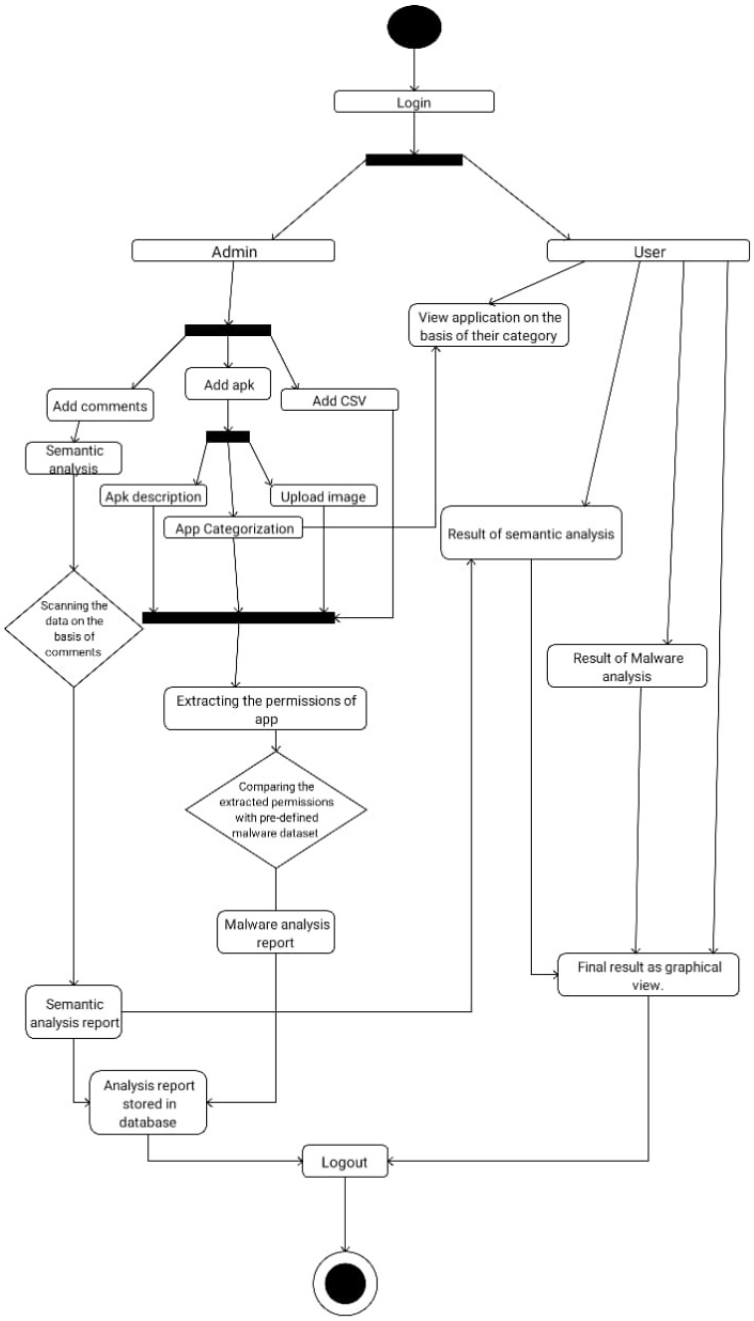


Figure 3.5: Activity Diagram

Activity Diagram depicts the flow of activity of our system.Starting with the registration user will have to login into the system.Further the user will interact with admin which includes requirement gathering.Based on this appropriate actions it is been given to the user.

Chapter 4

Project Implementation

Code Snippets

```
1
2  package com.appanalysis.service;
3
4  import java.io.BufferedReader;
8
9  public class AppAnalysis {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         String videotoimg = "C:\\Users\\RISHAB\\AppData\\Local\\Android\\Sdk\\build-tools\\29.0.2\\aapt d permissions ";
14         try {
15             Process p=Runtime.getRuntime().exec(videotoimg);
16             InputStream is = p.getInputStream();
17             InputStreamReader isr = new InputStreamReader(is);
18             BufferedReader br = new BufferedReader(isr);
19             String line = null;
20             while((line=br.readLine())!=null)
21             {
22                 //System.out.println(line);
23                 if(line.contains("uses-permission"))
24                 {
25                     String words[]=line.split("\\'");
26                     System.out.println(words[1]);
27                 }
28             }
29         } catch (IOException e) {
30             // TODO Auto-generated catch block
31             e.printStackTrace();
32         }
33         System.out.println("before exitting");
34     }
35
36 }
37
```

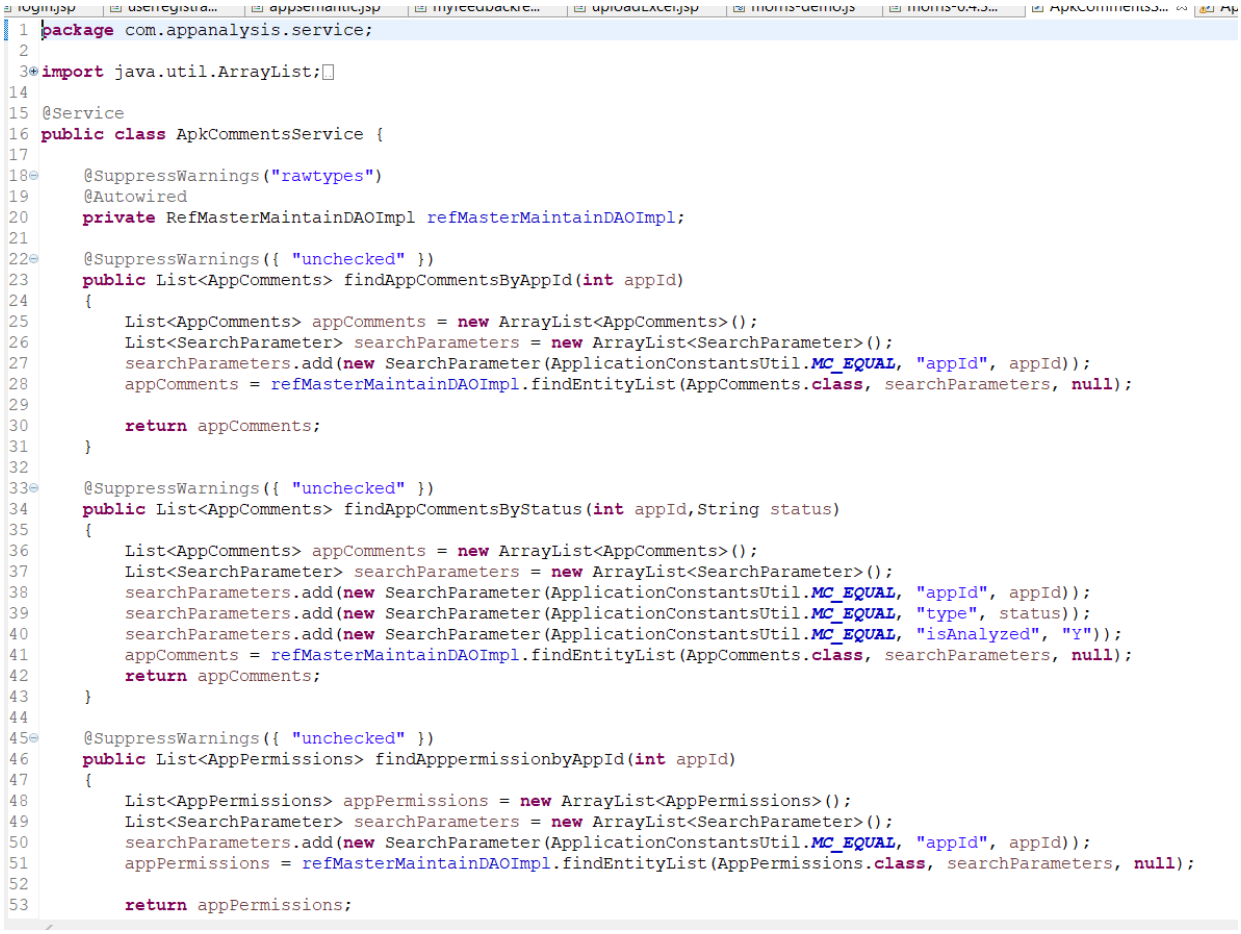
Figure 4.1: Permission Extraction

```
2
3  import java.util.ArrayList;
12
13 @Component
14 public class Helper {
15
16
17     private static List<String> maliciousPermission = new ArrayList<String>();
18
19     static{
20
21         maliciousPermission.add("android.permission.READ_PHONE_STATE");
22         maliciousPermission.add("com.google.android.c2dm.permission.RECEIVE");
23         maliciousPermission.add("android.permission.BLUETOOTH");
24
25     }
26
27     public static List<String> getMaliciousPermission()
28     {
29         return maliciousPermission;
30     }
31
32
33     public static String getUsername()
34     {
35         final Authentication authentication = SecurityContextHolder
36                 .getContext().getAuthentication();
37         if ((authentication == null)
38                 || authentication.getPrincipal().equals("anonymousUser"))
39         {
40             return null;
41         }
42         else
43         {
44             return authentication.getName();
45         }
46
47     }
48
49     public  static String generatePin() throws Exception {
50         final Random generator = new Random();
51         generator.setSeed(System.currentTimeMillis());
52         String genCode = Integer.toString(generator.nextInt(899999) + 100000);
```

Figure 4.2: Malware Dataset

```java
package com.appanalysis.service;

import java.util.ArrayList;

@Service
public class SemanticAnalysisService {

    @SuppressWarnings("rawtypes")
    @Autowired
    private RefMasterMaintainDAOImpl refMasterMaintainDAOImpl;

    @Scheduled(fixedDelay=6000000)
    public void performSemanticAnalysis()
    {
        List<AppComments> appcomments = this.getPendingComments();
        if(appcomments!=null && !appcomments.isEmpty())
        {
            for(AppComments a: appcomments)
            {
                SentimentAnalyzer sentimentAnalyzer = new SentimentAnalyzer();
                sentimentAnalyzer.initialize();
                SentimentResult sentimentResult = sentimentAnalyzer.getSentimentResult(a.getComment());
                a.setIsAnalyzed("Y");
                a.setType(sentimentResult.getSentimentType());
                a.setScore(String.valueOf(sentimentResult.getSentimentScore()));
                try {
                    refMasterMaintainDAOImpl.saveOrUpdate(a);
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
        }
    }

    private List<AppComments> getPendingComments()
    {
        AppComments appDetails = new AppComments();
        List<SearchParameter> splist = new ArrayList<SearchParameter>();
        splist.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "isAnalyzed", "N"));
        List<AppComments> appCommentsList = refMasterMaintainDAOImpl.findEntityList(AppComments.class, splist, null);
        return appCommentsList;
```

Figure 4.3: Semantic Analysis



```java
package com.appanalysis.service;

import java.util.ArrayList;

@Service
public class ApkCommentsService {

    @SuppressWarnings("rawtypes")
    @Autowired
    private RefMasterMaintainDAOImpl refMasterMaintainDAOImpl;

    @SuppressWarnings({ "unchecked" })
    public List<AppComments> findAppCommentsByAppId(int appId)
    {
        List<AppComments> appComments = new ArrayList<AppComments>();
        List<SearchParameter> searchParameters = new ArrayList<SearchParameter>();
        searchParameters.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "appId", appId));
        appComments = refMasterMaintainDAOImpl.findEntityList(AppComments.class, searchParameters, null);

        return appComments;
    }

    @SuppressWarnings({ "unchecked" })
    public List<AppComments> findAppCommentsByStatus(int appId,String status)
    {
        List<AppComments> appComments = new ArrayList<AppComments>();
        List<SearchParameter> searchParameters = new ArrayList<SearchParameter>();
        searchParameters.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "appId", appId));
        searchParameters.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "type", status));
        searchParameters.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "isAnalyzed", "Y"));
        appComments = refMasterMaintainDAOImpl.findEntityList(AppComments.class, searchParameters, null);
        return appComments;
    }

    @SuppressWarnings({ "unchecked" })
    public List<AppPermissions> findApppermissionbyAppId(int appId)
    {
        List<AppPermissions> appPermissions = new ArrayList<AppPermissions>();
        List<SearchParameter> searchParameters = new ArrayList<SearchParameter>();
        searchParameters.add(new SearchParameter(ApplicationConstantsUtil.MC_EQUAL, "appId", appId));
        appPermissions = refMasterMaintainDAOImpl.findEntityList(AppPermissions.class, searchParameters, null);

        return appPermissions;
```

Figure 4.4: Graph Service

1.ADMIN LOGIN.

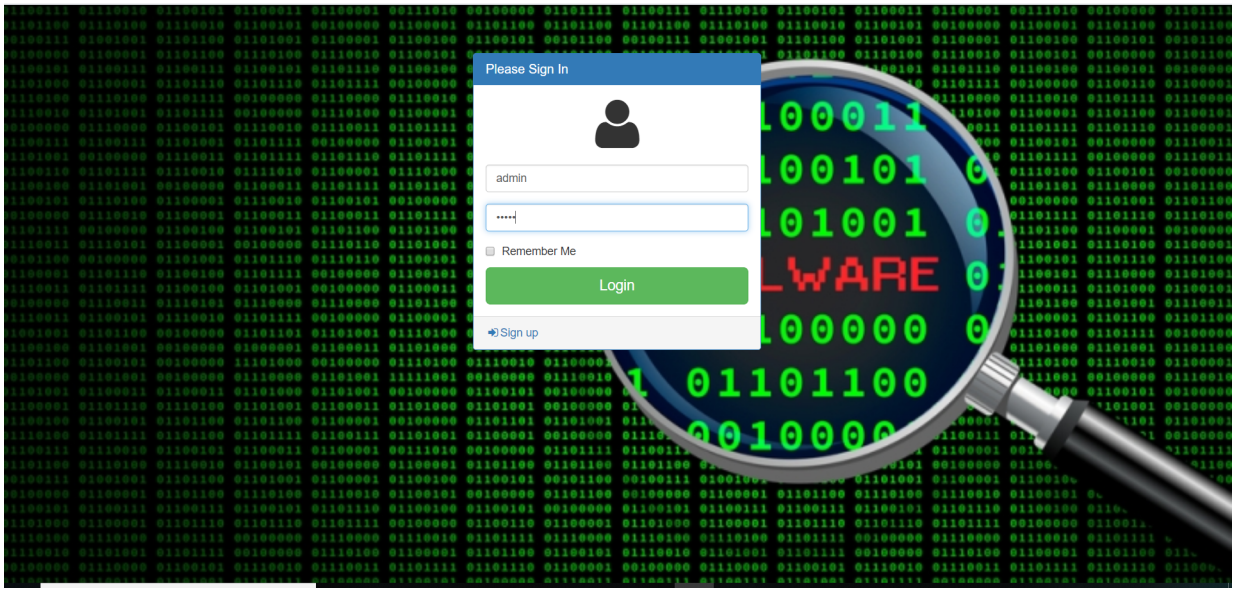


Figure 4.5: Admin login

In this the Admin login starts with the Id and the Particular password provided to you.

2.ADMIN HOME PAGE AFTER LOGIN.

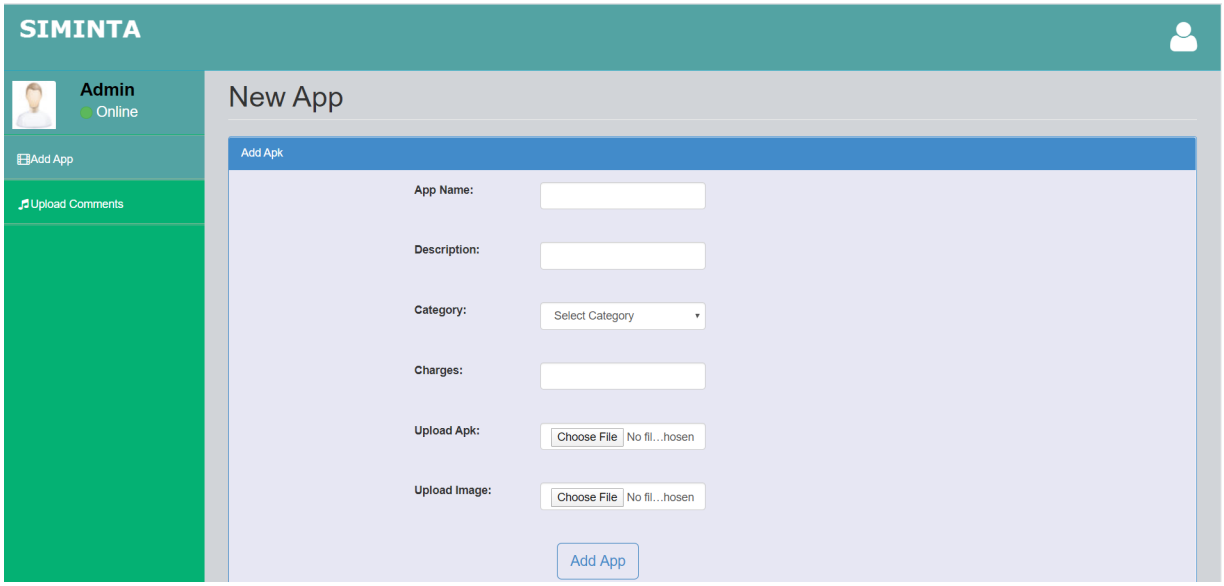


Figure 4.6: Admin home page after login

3.UPLOAD APK AND ENTER DETAILS.

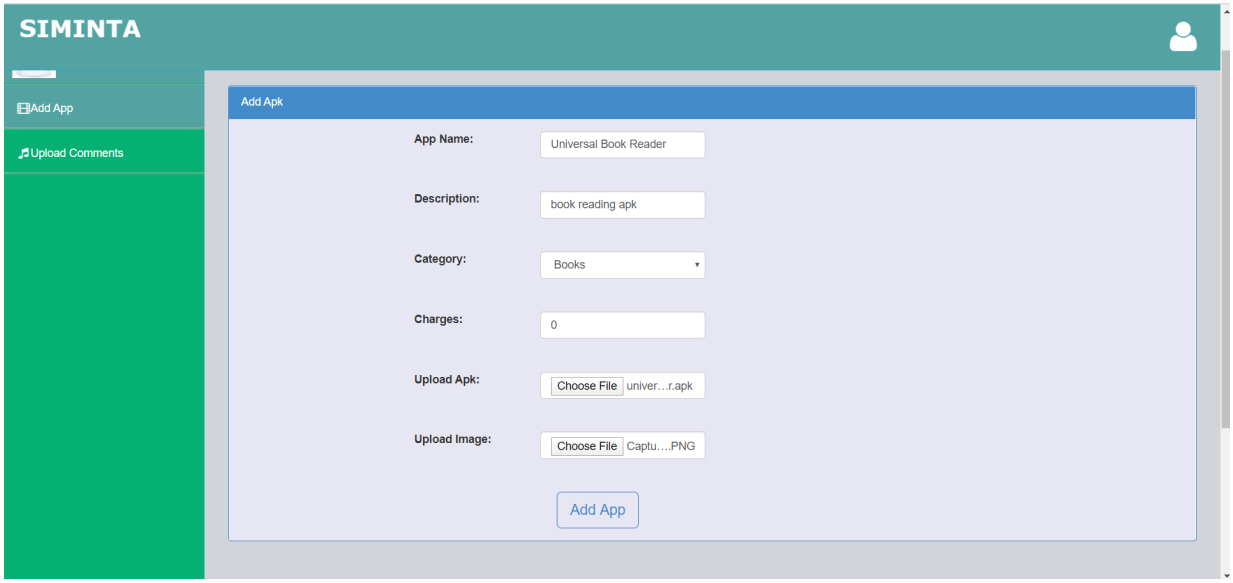


Figure 4.7: Add APK

In this the Admin needs to add an Apk file and also fills the details of the Apk files with the name, discription, category and addying an logo or image for an app and clicks on Add Apk.

4.UPLOAD COMMENTS.

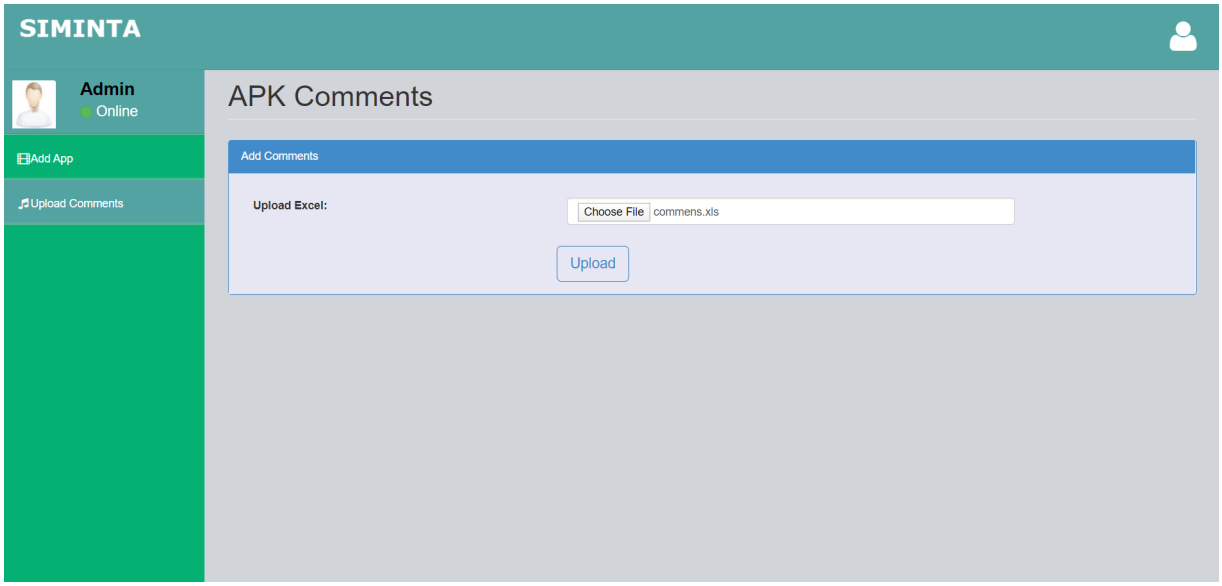


Figure 4.8: Upload Comments

Here the Comments are been uploaded of the particular Apk file which we uploaded. In this the CSV file is been used so that the comments are uploaded in that proper format.

5.USER PORTAL SIGNUP.

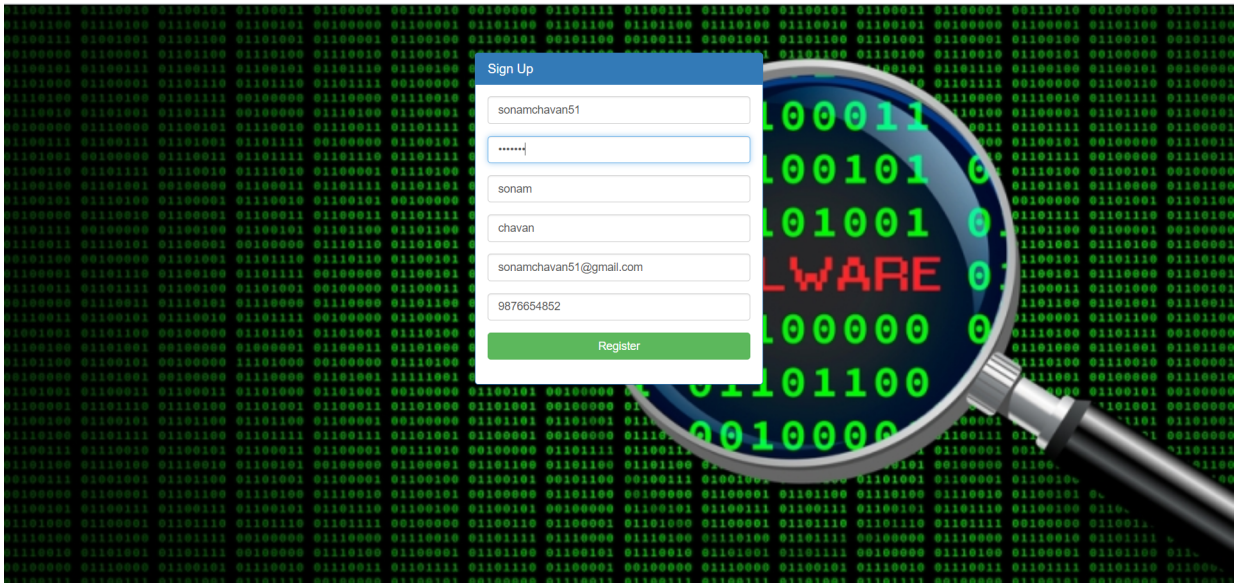


Figure 4.9: User Signup

The user is able to register himself with the help of this with providing the proper details of the user.

6.NEW USER CODE ACTIVATION DURING SIGNUP

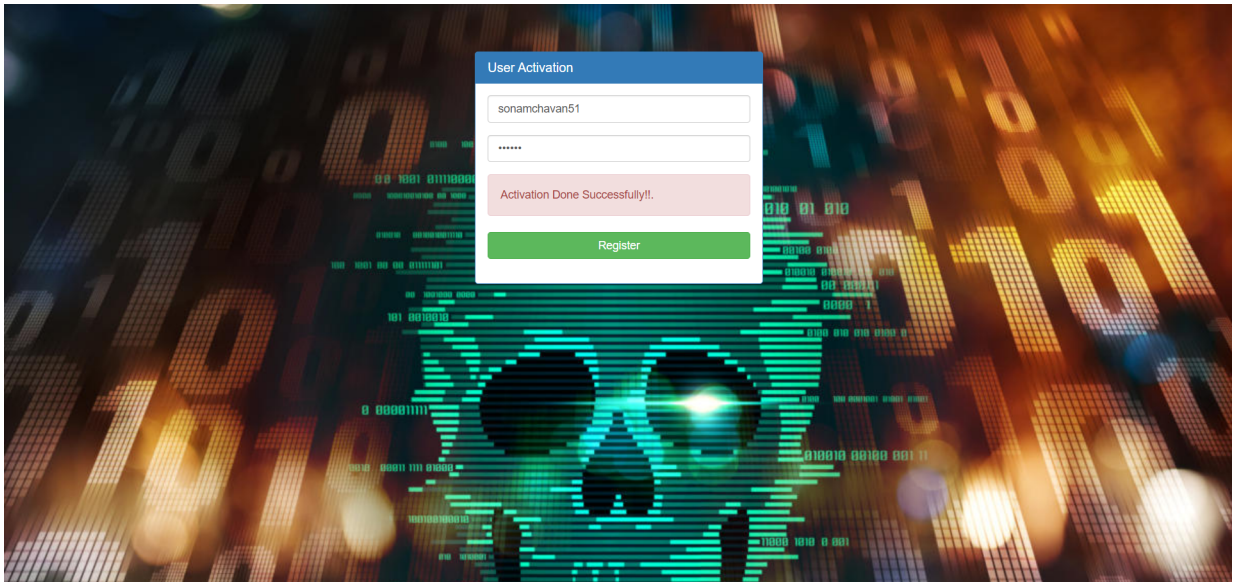


Figure 4.10: User Code Activation

In this the new user who signup will get a code over email or message system and it is an sort of verification being done.

7.USER PANEL.

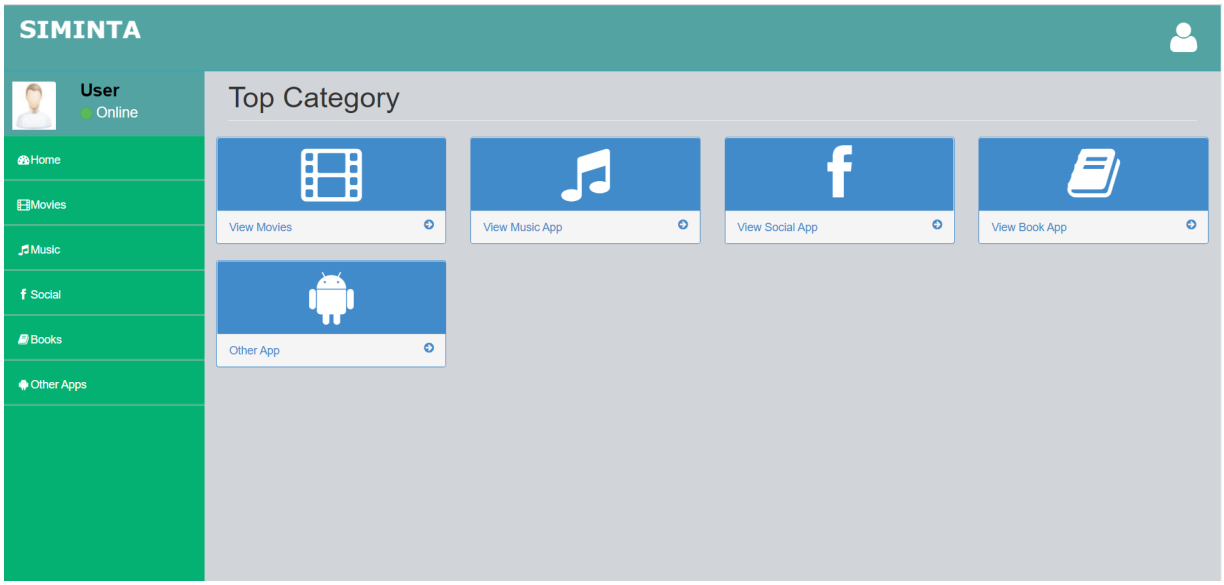


Figure 4.11: User Panel

In the User Panel we would be able to see the different categories of the app on that whichever we clicks the app which is been uploaded by the admin will be seen in that categories.

8.SOCIAL APP CATEGORY

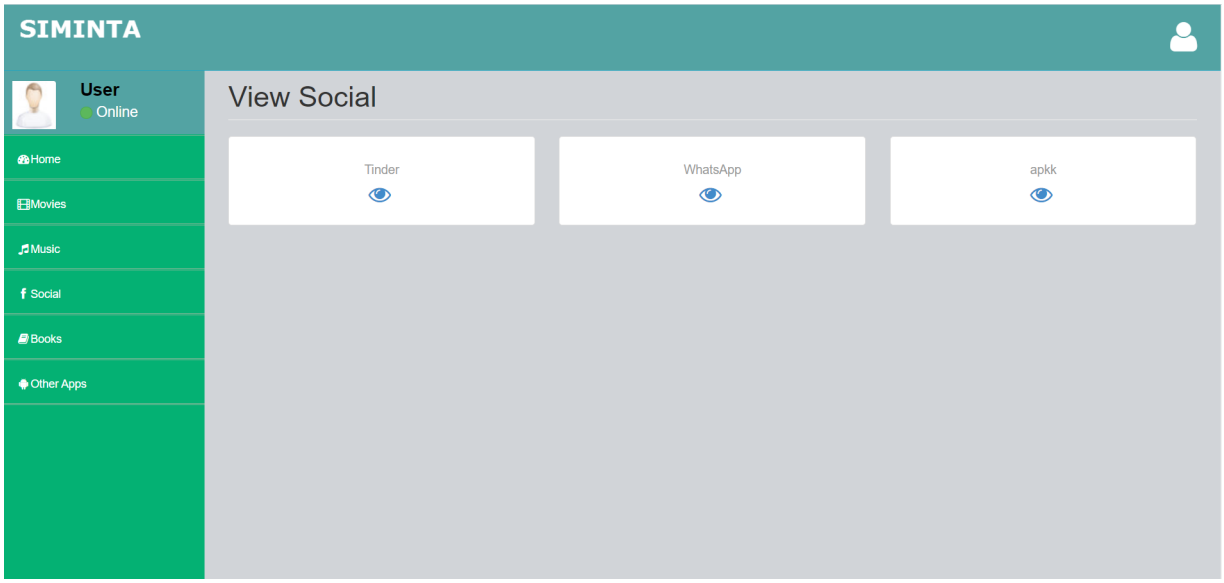


Figure 4.12: Social Apps

In the Social App Category the social apps which are been added by the admin side is been seen.

9.UPLOADED COMMENTS IN AN APP.

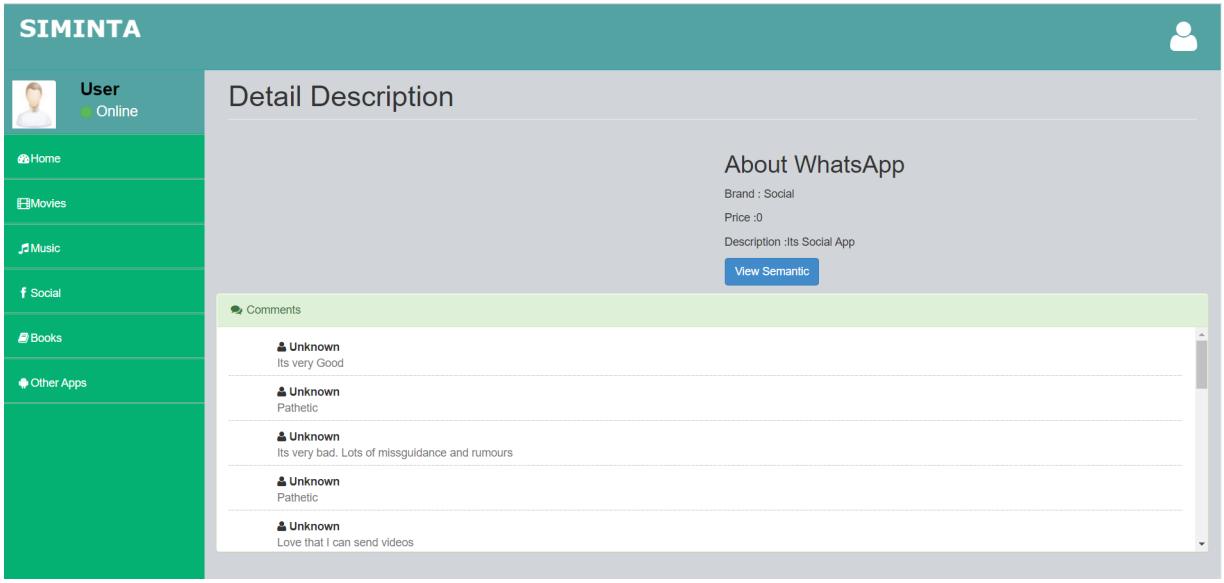


Figure 4.13: Comments in an App

In this the comments which we have been uploaded in the admin side under that app name as we click on it the comments are been shown of that app.

10.MALLICIOUS RESULTS

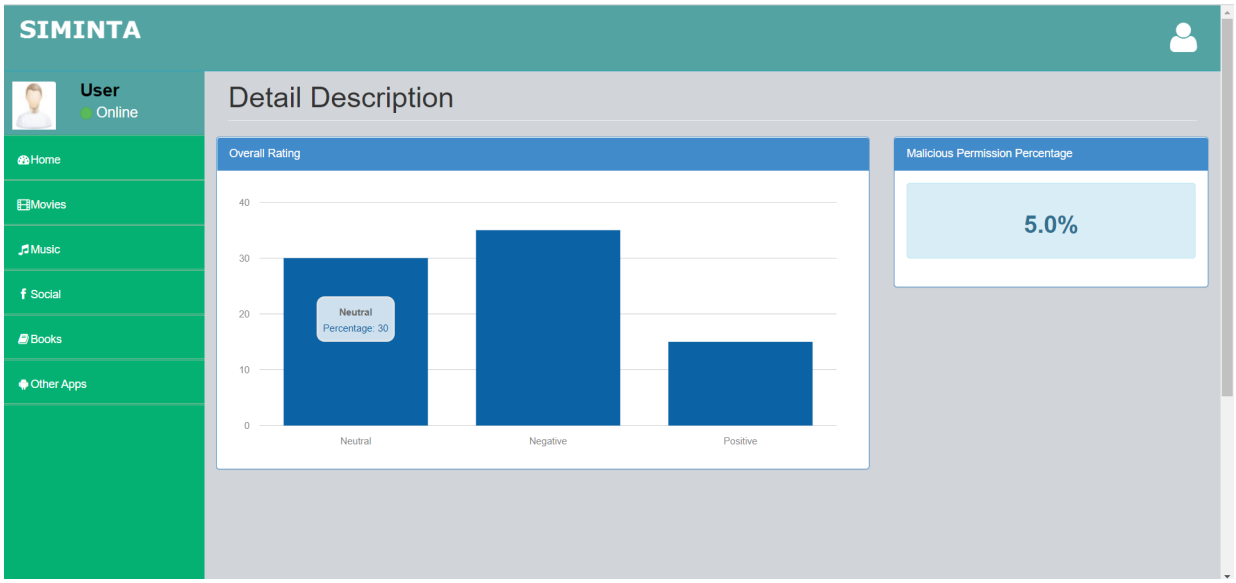


Figure 4.14: Mallicious Results in an App

The results are based on our dataset and a graph is been shown where the app is perfectly able to use by the user or not is been shown with the help of graph.

Chapter 5

Testing Result

Software testing methodologies are the various strategies or approaches used to test an application to ensure it behaves and looks as expected. These encompass everything from front to back-end testing, including unit and system testing.
The types of testing methods are:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

So in our testing methods we are using Unit Testing and Integrate Testing.

Unit Testing: Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process.

Integration Testing: Integration ttesting is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

5.1 Test Plan

- To achieve the problem statement.
- Performing tests until all test cases are passed.

Results for Functional Testing					
Test ID	**Test Cases**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Redirection of Pages	Click on any button	Should redirect to the next page depending upon the action called	Gets redirected successfully	Passed
2	Add an APK	Check for all the action buttons	Every button must perform certain action	Every button is specified with a certain action	Passed
3	Proper data input	Enter incorrect data while logging	Should not login giving an error	User not registered	Passed
4	Add comments	Enter proper comments file	Should take the comments file	Comments to be added	Passed

Table 5.1: Functional Testing

Results for Database Testing					
Test ID	**Test Cases**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Correct login with credentials	Enter correct login credentials	Successful Login	Login Successful	Passed
2	Incorrect login credentials	Enter incorrect login credentials	Unsuccessful login	Login Unsuccessful	Passed
3	Uploading the apk file	Select the file to upload	File should be selected and uploaded	File selected and uploaded	Passed
4	Upload another file format	Select the file to upload	File should not be uploaded	File selected cannot be upload	Passed
5	Upload CSV file	Select the particular file to upload	File should be selected and uploaded	File selected and uploaded	Passed

Table 5.2: Database Testing

Chapter 6

Conclusion and Future scope

6.1 Conclusion

In our work, we propose a system for permission analysis and semantic analysis. Our system is also used to detect malware permissions which an application may ask during or after installing. The permissions asked by the application are compared with a dataset present in the proposed system and the result is given out in the basis of permission analysis. In the semantic analysis, the system will give the malware report on the basis of reviews and ratings given to the particular application by the users in real time scenario. This proposed system can be applied in the fields of the security system and also for the other users like a malware detection software.

6.2 Future Scope

However, there are limitations in our system. The permissions which we are defining are as per our but it can differ from users to users. The permissions which the user likes that it is not a malware-based can be malware for any other user. Future works will contain the improvement of that.

Bibliography

[1] Darus,Fauzi Mohd,Salleh Noor Azurati Ahmad,and Aswami Fadillah Mohd Ariffin."Android Malware Detection Using Machine Learning on Image Patterns"2018 Cyber Resilience Conference(CRC).IEEE,2018.

[2] Vrama,P.Ravi Kiran,Kotari Prudvi raj,and KV Subba Raju."Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms."2017 InternationalConference on I-SMAC (IoT in Social,Mobile,Analytics and Cloud)(I-SMAC).IEEE2017.

[3] Chang,Wei-Ling, Hung-Min Sun,and Wei Wu."An Android BehaviourBased Malware Detection Method using Machine Learning." 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC).IEEE, 2016.

[4] Koli, J. D. "RanDroid: Android malware detection using random machine learning classifiers." 2018 Technologies for Smart-City Energy Security and Power (ICSESP).IEEE,2018.

Acknowledgement

We have great pleasure in presenting the report on **Android Malware Detection Using Machine Learning.** We take this opportunity to express our sincere thanks towards our guide **Mr. Ganesh Gourshete** & Co-Guide **Ms. Nahid Shaikh** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Mr. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Mr. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Rishab Agrawal

17204006

Vishal Shah

17204003

Sonam Chavan

16104067

Publication

Paper entitled **"Android Malware Detection using Machine Learning"** is presented at **"IEEE Xplore in the conference ic-ETITE(2020 International Conference on Emerging Trends in Information Technology and Engineering)"** by **"Rishab Agrawal" , "Vishal Shah" , "Sonam Chavan" , "Prof. Ganesh Gourshete" , "Prof. Nahid Shaikh" .**