

A PROJECT REPORT ON
"DETECTING HARMFUL MOBILE WEBPAGES"

submitted in partial fulfilment of requirement
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY

NAME	REGISTER NUMBER
P.SAI PRIYANKA	15091A05C5
V.YAMINI SARASWATHI	15091A05H3
G.V.N. SRAVYA	15091A05F9
P.VIKAS PAUL	15091A05G7
P.SAI TEJASWANI	15091A05C6

Under the guidance of
DR. R. RAJA KUMAR

Associate Professor in CSE Department



(ESTD-1995)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING &
TECHNOLOGY**
(AUTONOMOUS)

Accredited by NAAC of UGC, New Delhi with 'A⁺' Grade
Approved by AICTE, Affiliated to J.N.T. University Anantapur, Nandyal-518501

2018-2019

Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)

Professor & Head of the Department of CSE

RGM College of Engg. & Tech., (Autonomous)

NANDYAL-518 501, Kurnool (Dist), A.P.

**RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING AND
TECHNOLOGY
(AUTONOMOUS)**

Accredited by NAAC of UGC, New Delhi with 'A⁺' Grade
Approved by AICTE, Affiliated to J.N.T. University Anantapur, Nandyal-518501



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that that P. Sai Priyanka (15091A05C5), V. Yamani Saraswathi (15091A05H3), G.V.N.Sravya(15091A05F9), P. Vikas Paul (15091A05G7), P.Sai Tejaswani (15091A05C6), of B.Tech (CSE) final year students has carried out the project work on "DETECTING HARMFUL MOBILE WEBPAGES" under the esteemed guidance of Dr. R. Raj Kumar, Associate Professor in CSE Department, for the partial fulfilment of the award of degree of B.Tech (CSE) in RGM CET, Nandyal as a bonafide record of work done by them during the year 2018-2019.

Head of the department

Dr. K. SubbaReddy M.Tech, Ph.D.
Associate Professor
Department of CSE

Project Guide

Dr. R. Raja Kumar, M.Tech, Ph.D.
Associate Professor
Department of CSE

Place: Nandyal

Examiner:

Date :

Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurnool (Dist), A.P.

ACKNOWLEDGEMENT

We manifest our heartier thankfulness pertaining to your contentment over our project guide **Dr. R. Raja Kumar**, M.Tech, Ph.D., Associate Professor in Computer science and Engineering department, with whose adroit concomitance the excellence has been exemplified in bringing out this project to work with artistry.

We express our gratitude to **Dr. K. Subba Reddy garu**, Head of the Department of Computer science Engineering department, all teaching and non-teaching staff of the Computer science Engineering department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of our project.

At the outset we thank our honourable **Chairman Dr. M. Santhi Ramudu garu**, for providing us with exceptional faculty and moral support throughout the course.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr.T. Jaya Chandra Prasad garu**, who has been observed posing valiance in abundance towards our individuality to acknowledge our project work tangentially.

Finally we extend our sincere thanks to all the **Staff Members** of CSE Department who have co-operated and encouraged us in making our project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. We see in every good that happens to us their love and blessings.

By:

P.SAI PRIYANKA	(15091A05C5)
V.YAMINI SARASWATHI	(15091A05H3)
G.V.N SRAVYA	(15091A05F9)
P.VIKAS PAUL	(15091A05G7)
P.SAI TEJASWANI	(15091A05C6)

Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous),
NANDYAL-518 501 Kurnool (Dist), A.P.

CANDIDATE'S DECLARATION

We here by declare that the work done in this Project titled "**DETECTING HARMFUL MOBILE WEBPAGES**" submitted towards completion of project work in IV Year II Semester of B.Tech (CSE) at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record our original work done under the guidance of **DR. R. Raja Kumar, Associate Professor**, Dept. of CSE, RGM CET, Nandyal. We have not submitted the matter embodied in this project for the award of any other Degree in any other institutions.

By:

P.SAI PRIYANKA	(15091A05C5)
V.YAMINI SARASWATHI	(15091A05H3)
G.V.N SRAVYA	(15091A05F9)
P.VIKAS PAUL	(15091A05G7)
P.SAI TEJASWANI	(15091A05C6)

Place: Nandyal

Date:

14

Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engineering & Tech., (Autonomous)
NANDYAL, Nandyal (Dist), A.P.

ABSTRACT

Mobile specific webpages differ significantly from their desktop counterparts in content, layout and functionality. Accordingly, existing techniques to detect malicious websites are unlikely to work for such webpages. In this, we present KAYO, a mechanism that distinguishes between malicious and benign mobile web pages. It is a fast and reliable static analysis technique to detect malicious mobile web-pages. KAYO uses static features of mobile webpages derived from their HTML and JavaScript content, URL and advanced mobile specific capabilities. First, we experimentally demonstrate the need for mobile specific techniques and then identify a range of new static features that highly correlate with mobile malicious webpages. Finally, we build a browser extension using KAYO to protect users from malicious mobile websites in real-time. In doing so, we provide the first static analysis technique to detect malicious mobile webpages.



Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)

Professor & Head of the Department of CSE
RGM College of Engg & Tech., (Autonomous)
NANDYAL-518 301 (Dist), A.P.

CONTENTS

List of Figures

List of Abbreviations

Chapter 1: Introduction 1-5

Chapter 2: Literature Survey 6-8

Chapter 3: System Analysis 9-10

3.1 Existing System

3.2 Disadvantages of Existing System

3.3 Proposed System

3.4 Advantages of Proposed System

3.5 Hardware Requirements

3.6 Software Requirements

Chapter 4: Feasibility Study 11-12

4.1 Technical Feasibility

4.2 Operational Feasibility

4.3 Economic Feasibility

Chapter 5: System Design 13-26

5.1 Modules

5.2 System Architecture

5.3 UML (Unified Modelling Language)

5.4 List of Symbols

5.5 UML Diagrams

Chapter 6: Implementation 27-39

6.1 Technical Description

14
Dr. SUBBA REDDY KUNAM

BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 501, Kurunot (malk) A.P.

Chapter 7: Testing	40-42
7.1 Testcase description	
7.2 Levels of Testing	
7.3 Types of Testing	
Chapter 8: Screenshots	43-51
Chapter 9: Conclusion	52
REFERENCES	53


Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
NANDYAL-518 001, Nellore (Dist), A.P.

LIST OF FIGURES

FIGURES	Page No
5.1 System Architecture diagram	14
5.2 Flow Chart – admin	15
5.3 Flow Chart – User	16
5.4 Use Case Diagram	22
5.5 Class Diagram	23
5.6 Sequence Diagram	24
5.7 Data Flow	25

14
Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
Narasimharao Nagar, Kurnool (Dist), A.P.

LIST OF ABBREVIATIONS

- HTML - Hyper Text Markup Language
CSS - Cascading Style Sheet
JVM - Java Virtual Machine
URL - Uniform Resource Locator
UML - Unified Modeling Language
JDBC - Java Database Connectivity
ODBC - Open Database connectivity

14
Dr. SUBBA REDDY KUNAM
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)
Professor & Head of the Department of CSE
RGM College of Engg. & Tech., (Autonomous)
Narasimhalu-518 501, Kurnool (Dist), A.P.

CHAPTER 1

INTRODUCTION

Mobile devices are increasingly being used to access the web. However, in spite of significant advances in processor power and bandwidth, the browsing experience on mobile devices is considerably different. These differences can largely be attributed to the dramatic reduction of screen size, which impacts the content, functionality and layout of mobile webpages.

Content, functionality and layout have regularly been used to perform static analysis to determine maliciousness in the desktop space. Features such as the frequency of iframes and the number of redirections have traditionally served as strong indicators of malicious intent. Due to the significant changes made to accommodate mobile devices, such assertions may no longer be true. For example, whereas such behavior would be flagged as suspicious in the desktop setting, many popular benign mobile webpages require multiple redirections before users gain access to content. Previous techniques also fail to consider mobile specific webpage elements such as calls to mobile APIs. For instance, links that spawn the phone's dialer (and the reputation of the number itself) can provide strong evidence of the intent of the page. New tools are therefore necessary to identify malicious pages in the mobile web.

In this, we present kAYO, a fast and reliable static analysis technique to detect malicious mobile webpages. kAYO uses static features of mobile webpages derived from their HTML and JavaScript content, URL and advanced mobile specific capabilities. We first experimentally demonstrate that the distributions of identical static features when extracted from desktop and mobile webpages vary dramatically. kAYO's performance matches or exceeds that of existing static techniques used in the desktop space. kAYO also detects a number of malicious mobile webpages not precisely detected by existing techniques such as VirusTotal and Google Safe Browsing. Finally, we discuss the limitations of existing tools to detect mobile malicious webpages and build a browser extension based on kAYO that provides realtime feedback to mobile browser users.

Experimentally demonstrate the differences in the "security features" of desktop and mobile webpages:

Experimentally demonstrate that the distributions of static features used in existing techniques (e.g., the number of redirections) are different when measured on mobile and

desktop webpages. Moreover, we illustrate that certain features are inversely correlated or unrelated to or non-indicative to a webpage being malicious when extracted from each space. The results of experiments demonstrate the need for mobile specific techniques for detecting malicious webpages.

Design and implement a classifier for malicious and benign mobile webpages:

We then identify new static features that distinguish between mobile benign and in classification and shows improvement of two orders of magnitude in the speed of feature extraction over similar existing techniques. We further empirically demonstrate the significance of kAYO's features. Finally, we also identify mobile webpages implementing cross-channel attacks, which attempt to induce mobile users to call numbers associated with known fraud campaigns.

Implement a browser extension based on kAYO:

To the best of our knowledge kAYO is the first technique that detects mobile specific malicious webpages by static analysis. Existing tools such as Google Safe Browsing are not enabled on the mobile versions of browsers, thereby precluding mobile users. Moreover, the mobile specific design of kAYO enables detection of malicious mobile webpages missed by existing techniques. Finally, our survey of existing extensions on Firefox desktop browser suggests that there is a paucity of tools that help users identify mobile malicious webpages. To fill this void, we build a Firefox mobile browser extension using kAYO, which informs users about the maliciousness of the webpages they intend to visit in real-time. We plan to make the extension publicly available post publication.

Content-based and in-depth inspection techniques to detect malicious websites:

Dynamic approaches using virtual machines and honey client systems provide deeper visibility into the behavior of a webpage. Therefore, such systems have a very low false positive rate and are more accurate. However, downloading and executing each webpage impacts performance and hinders scalability of dynamic approaches. This performance penalty can be avoided by using static approaches. Static approaches rely on the structural and lexical properties of a webpage and do not execute the content of the webpage. One such technique of detecting malicious URLs is using statistical methods for URL classification based on a URL's lexical and host-based properties. However, URL-based techniques usually suffer from high false positive rates. Using HTML and JavaScript features extracted from a

webpage in addition to URL classification helps address this drawback and provides better results. Static approaches avoid performance penalty of dynamic approaches. Additionally, using fast and reliable static approaches to detect benign webpages can avoid expensive in-depth analysis of all webpages.

Differences between mobile and desktop websites:

All these approaches for malicious webpage detection have focused on websites built for desktop browsers in the past. Mobile browsers have been shown to differ from their desktop counterparts in terms of security. Although differences in mobile and desktop websites have been observed before, it is unclear how these differences impact security. Furthermore, the threats on mobile and desktop websites are somewhat different. Static analysis techniques using features of desktop webpages have been primarily studied for drive-by-downloads on desktop websites, whereas, the biggest threat on the mobile web at present is believed to be phishing. Efforts in mitigating phishing attacks on desktop websites include isolating browser applications of different trust level, email filtering, using content-based features and blacklists. The best-known non-proprietary content-based approach to detect phishing webpages is Cantina. Cantina suffers from performance problems due to the time lag involved in querying the Google search engine. Moreover, Cantina does not work well on webpages written in languages other than English. Finally, existing techniques do not account for new mobile threats such as known fraud phone numbers that attempt to trigger the dialer on the phone. Consequently, whether existing static analysis techniques to detect malicious desktop websites will work well on mobile websites is yet to be explored.

Mobile application security:

Significant work has been done in the past few years on the security of mobile applications. Static feature extraction, especially with respect to permissions, has been one of the most important early areas of research. Such techniques have led to dramatically more rapid detection of malicious applications across a range of marketplaces.

DNS based approaches to detect malicious domains:

A popular approach in detecting malicious activity on the web is by leveraging distinguishing features between malicious and benign DNS usage. Both passive DNS monitoring and active DNS probing methods, have been used to identify malicious domains. While some of these efforts focused solely on detecting fast flux service networks, another

can also detect domains implementing phishing and drive-by-downloads. DNS based mechanisms do not provide deeper understanding of the specific activity implemented by a webpage or domain.

Static analysis techniques to detect malicious websites often use features of a webpage such as HTML, JavaScript and characteristics of the URL. Usually, these features are fed to machine learning techniques to classify benign and malicious webpages. These techniques are predicated on the assumption that the features are distributed differently across benign and malicious webpages. Accordingly, any changes in the distribution of static features in benign and/or malicious webpages impacts classification results of static analysis techniques. While successful, these static analysis techniques have been used exclusively for desktop webpages. Mobile websites are significantly different from their desktop counterparts in content, functionality and layout. Consequently, existing tools using static features to detect malicious desktop webpages are unlikely to work for mobile webpages. We explain four factors that motivate building separate static analysis techniques to detect malicious mobile webpages.

Differences in content:

Mobile websites are often simpler than their desktop counterparts. Therefore, the distribution of content-based static features (such as the number of JavaScripts) on mobile webpages differs from that of desktop webpages. Approximately 90% of mobile webpages do not have any iframes, whereas the corresponding desktop webpages have multiple iframes. Desktop webpages have more JavaScripts than mobile webpages. Due to the simplicity of mobile webpages, the majority of other content related static features used in existing techniques including, the number of images, page length, the number of hidden elements, and the number of elements with a small area also differ in magnitude in mobile and desktop webpages.

Infrastructure:

Website providers use JavaScript or user agent strings to identify and then redirect mobile users to a mobile specific version. Even the most popular mobile websites show multiple redirects, which has traditionally been a property of desktop websites hosting malware. However, multiple redirects does not necessarily indicate bad behavior for mobile websites due to the characteristics of their hosting infrastructure. We note that not all static features used in existing techniques differ when measured on mobile and desktop webpages.

For example, the number of IP addresses returned by DNS servers for mobile and desktop versions of the same sites is comparable. Mobile websites appear to share their hosting infrastructure with the corresponding desktop websites. We use the term anal URL to denote the URL that is rendered in the browser after redirections (if any) from the seed URL. The anal URL may change based on the browser's user agent for mobile and desktop websites.

Impact of screen size:

The screen size of a mobile phone is significantly smaller than that of a desktop computer. Therefore, a mobile user only sees a part of the URL of a webpage. Intuitively, the author of a mobile phishing webpage may only need to include misleading words at the beginning of the URL and a short URL might suffice to trick a user.

Mobile specific functionality:

Mobile websites enable access to a user's personal information and advanced capabilities of mobile devices through web APIs. Existing static analysis techniques do not consider these mobile specific functionalities in their feature set. We argue and later demonstrate that accounting for the mobile specific functionalities helps identify new threats specific to the mobile web. For example, the presence of a known 'bank' fraud number on a website might indicate that the webpage is a phishing webpage imitating the same bank .

CHAPTER 2

LITERATURE SURVEY

1) Detecting malicious websites with low-interaction honeyclients

AUTHORS: A. Ikinici, T. Holz, and F. Freiling. Monkey-spider:

Client-side attacks are on the rise: malicious websites that exploit vulnerabilities in the visitor's browser are posing a serious threat to client security, compromising innocent users who visit these sites without having a patched web browser. Currently, there is neither a freely available comprehensive database of threats on the Web nor sufficient freely available tools to build such a database. In this work, we introduce the Monkey-Spider project [Mon]. Utilizing it as a client honeypot, we portray the challenge in such an approach and evaluate our system as a high-speed, Internet scale analysis tool to build a database of threats found in the wild. Furthermore, we evaluate the system by analyzing different crawls performed during a period of three months and present the lessons learned.

2) A guided approach to finding malicious web pages

AUTHORS: L. Invernizzi, S. Benvenuti, M. Cova, P. M. Comparetti, C. Kruegel, and G. Vigna. Evilseed

Malicious web pages that use drive-by download attacks or social engineering techniques to install unwanted software on a user's computer have become the main avenue for the propagation of malicious code. To search for malicious web pages, the first step is typically to use a crawler to collect URLs that are live on the Internet. Then, fast prefiltering techniques are employed to reduce the amount of pages that need to be examined by more precise, but slower, analysis tools (such as honey clients). While effective, these techniques require a substantial amount of resources. A key reason is that the crawler encounters many pages on the web that are benign, that is, the "toxicity" of the stream of URLs being analyzed is low. In this paper, we present EVILSEED, an approach to search the web more efficiently for pages that are likely malicious. EVILSEED starts from an initial seed of known, malicious web pages. Using this seed, our system automatically generates search engine queries to identify other malicious pages that are similar or related to the ones in the initial seed. By doing so, EVILSEED leverages the crawling infrastructure of search engines to

retrieve URLs that are much more likely to be malicious than a random page on the web. In other words EVILSEED increases the "toxicity" of the input URL stream. Also, we envision that the features that EVILSEED presents could be directly applied by search engines in their prefilters. We have implemented our approach, and we evaluated it on a large-scale dataset. The results show that EVILSEED is able to identify malicious web pages more efficiently when compared to crawler-based approaches.

3) Blog identification and splog detection.

AUTHORS:P. Kolari, T. Finin, and A. Joshi. Svms for the blogosphere

Weblogs, or blogs have become an important new way to publish information, engage in discussions and form communities. The increasing popularity of blogs has given rise to search and analysis engines focusing on the 'blogosphere'. A key requirement of such systems is to identify blogs as they crawl the Web. While this ensures that only blogs are indexed, blog search engines are also often overwhelmed by spam blogs (splogs). Splogs not only incur computational overheads but also reduce user satisfaction. In this paper we first describe our experiments on blog identification using Support Vector Machines (SVM). We compare results of using different feature sets and introduce new features for blog identification. We then report preliminary results on splog detection and identify future work.

4) Phishdef: Url names say it all.

AUTHORS:A. Le, A. Markopoulou, and M. Faloutsos.

Phishing is an increasingly sophisticated method to steal personal user information using sites that pretend to be legitimate. In this paper, we take the following steps to identify phishing URLs. First, we carefully select lexical features of the URLs that are resistant to obfuscation techniques used by attackers.

Second, we evaluate the classification accuracy when using only lexical features, both automatically and hand-selected, vs. when using additional features. We show that lexical features are sufficient for all practical purposes. Third, we thoroughly compare several classification algorithms, and we propose to use an online method (AROW) that is able to overcome noisy training data. Based on the insights gained from our analysis, we propose PhishDef, a phishing detection system that uses only URL names and combines the above three elements. PhishDef is a highly accurate method (when compared to state-of-the-art

approaches over real datasets), lightweight (thus appropriate for online and client-side deployment), proactive (based on online classification rather than blacklists), and resilient to training data inaccuracies (thus enabling the use of large noisy training data).

5) The core of the matter: Analyzing malicious traffic in cellular carriers.

AUTHORS:C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee

Much of the attention surrounding mobile malware has focused on the in-depth analysis of malicious applications. While bringing the community valuable information about the methods used and data targeted by malware writers, such work has not yet been able to quantify the prevalence with which mobile devices are actually infected. In this paper, we present the first such attempt through a study of the hosting infrastructure used by mobile applications. Using DNS traffic collected over the course of three months from a major US cellular provider as well as a major US noncellular Internet service provider, we identify the DNS domains looked up by mobile applications, and analyze information related to the Internet hosts pointed to by these domains. We make several important observations. The mobile malware found by the research community thus far appears in a minuscule number of devices in the network: 3,492 out of over 380 million (less than 0.0009%) observed during the course of our analysis. This result lends credence to the argument that, while not perfect, mobile application markets are currently providing adequate security for the majority of mobile device users. Second, we find that users of iOS devices are virtually identically as likely to communicate with known low reputation domains as the owners of other mobile platforms, calling into question the conventional wisdom of one platform demonstrably providing greater security than another. Finally, we observe two malware campaigns from the upper levels of the DNS hierarchy and analyze the lifetimes and network properties of these threats. We also note that one of these campaigns ceases to operate long before the malware associated with it is discovered suggesting that network-based countermeasures may be useful in the identification and mitigation of future threats

CHAPTER-3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM :

- A popular approach in detecting malicious activity on the web is by leveraging distinguishing features between malicious and benign DNS usage.
- Both passive DNS monitoring and active DNS probing methods have been used to identify malicious domains. While some of these efforts focused solely on detecting fast flux service networks, another can also detect domains implementing phishing and drive-by-downloads.
- The best-known non-proprietary content-based approach to detect phishing webpages is Cantina

3.2. DISADVANTAGES :

- Existing tools such as Google Safe Browsing are not enabled on the mobile versions of browsers, thereby precluding mobile users.
- DNS based mechanisms do not provide deeper understanding of the specific activity implemented by a webpage or domain.
- Downloading and executing each webpage impacts performance and hinders scalability of dynamic approaches.
- URL-based techniques usually suffer from high false positive rates.
- Cantina suffers from performance problems due to the time lag involved in querying the Google search engine. Moreover, Cantina does not work well on webpages written in languages other than English.
- Finally, existing techniques do not account for new mobile threats such as known fraud phone numbers that attempt to trigger the dialer on the phone.

3.3. PROPOSED SYSTEM :

- In this paper, we present kAYO, a fast and reliable static analysis technique to detect malicious mobile web-pages. kAYO uses static features of mobile webpages derived from their HTML and JavaScript content, URL and advanced mobile specific capabilities.
- We first experimentally demonstrate that the distributions of identical static features

when extracted from desktop and mobile webpages vary dramatically

- We experimentally demonstrate that the distributions of static features used in existing techniques(e.g., the number of redirections) are different when measured on mobile and desktop webpages. Moreover, we illustrate that certain features are inversely correlated or unrelated to or non-indicative to a webpage being malicious when extracted from each space.

3.4. ADVANTAGES OF PROPOSED SYSTEM :

- kAYO also detects a number of malicious mobile webpages not precisely detected by existing techniques such as VirusTotal and Google Safe Browsing.
- The results of our experiments demonstrate the need for mobile specific techniques for detecting malicious webpages.
- To the best of our knowledge kAYO is the first technique that detects mobile specific malicious webpages by static analysis.
- Moreover, the mobile specific design of Kayo enables detection of malicious mobile webpages missed by existing techniques.
- Finally, our survey of existing extensions on Firefox desktop browser suggests that there is a paucity of tools that help users identify mobile malicious webpages.

3.5. HARDWARE REQUIREMENTS:

- System : Pentium IV 3.4 GHz (Min)or Later versions.
- Hard Disk : 40 GB.
- Monitor : 14' Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 1 GB.(Min)

3.6. SOFTWARE REQUIREMENTS:

- Operating system : Windows Family.
- Coding Language : J2EE (JSP,Servlet,Java Bean)
- Data Base : MY Sql Server.
- Web Server : Tomcat 6.0

CHAPTER -4

FEASIBILITY STUDY

Feasibility study is an important phase in the software development process. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

- ☐ Technical Feasibility
- ☐ Operation Feasibility
- ☐ Economic Feasibility

4.1 Technical Feasibility:

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- ☐ Does the necessary technology exist to do what is suggested?
- ☐ Do the proposed equipment's have the technical capacity to hold the data required to use the new system?
- ☐ Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- ☐ Can the system be upgraded if developed?

4.2 Operational Feasibility:

User-friendly:

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

Reliability:

The package will pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system. Security: The web server and database server should be protected from hacking, virus etc.

Portability:

The application will be developed using standard open source software (Except Oracle) like

Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not rise. Availability: This software will be available always. Maintainability: The system called the wheels uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-Sql, which is the back-embles will not arise.

4.3 Economic Feasibility:

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any additional hardware or software.

CHAPTER 5

SYSTEM DESIGN

5.1 MODULES

5.1.1 Admin module

In this module, admin has to login with valid username and password. After login successful he can do some operations such as -- View all users and authorize and Add Topics with Topic name, URL, Desc(enc), Uses, URL Author, Launched year, attach Topic image, List all topics urls with ranking order by desc and rating order by desc, Set Limit to access malicious WebPages and view, List all Malicious WebPages (if admin name is null, publisher name is Hacker) with attacker names with date and time and IP Address, List all Malicious WebPages accessed user details with date and time and IP Address, Block Malicious WebPages accessed user if they cross access limit and view the same, View all recommended WebPages by other users, View all Web pages viewed users details with date and time and IP Address, View Topic ranks in chart, view no. of time accessed specified Malicious web page by particular user in the chart, View No. of blocked and Un blocked users in the chart

5.1.2 User

In this module, User should register before searching the Website contents. After registration successful the user can login by using valid user name and password. After Login successful the user will do some operations --- View profile, Search WebPages by content keyword - Display only topic name order by description and WebPages and then click on topic name to view all details (increase rank), and recommend to other users, click on web url to display webpage, View all other user recommended Web pages, View Top k web pages urls and view the details (increase rank).

5.1.3 Attacker

In attacker page attacker can add the malicious webpages which looks same as normal webpages.

5.2 System Architecture

5.2.1 System Architecture Diagram

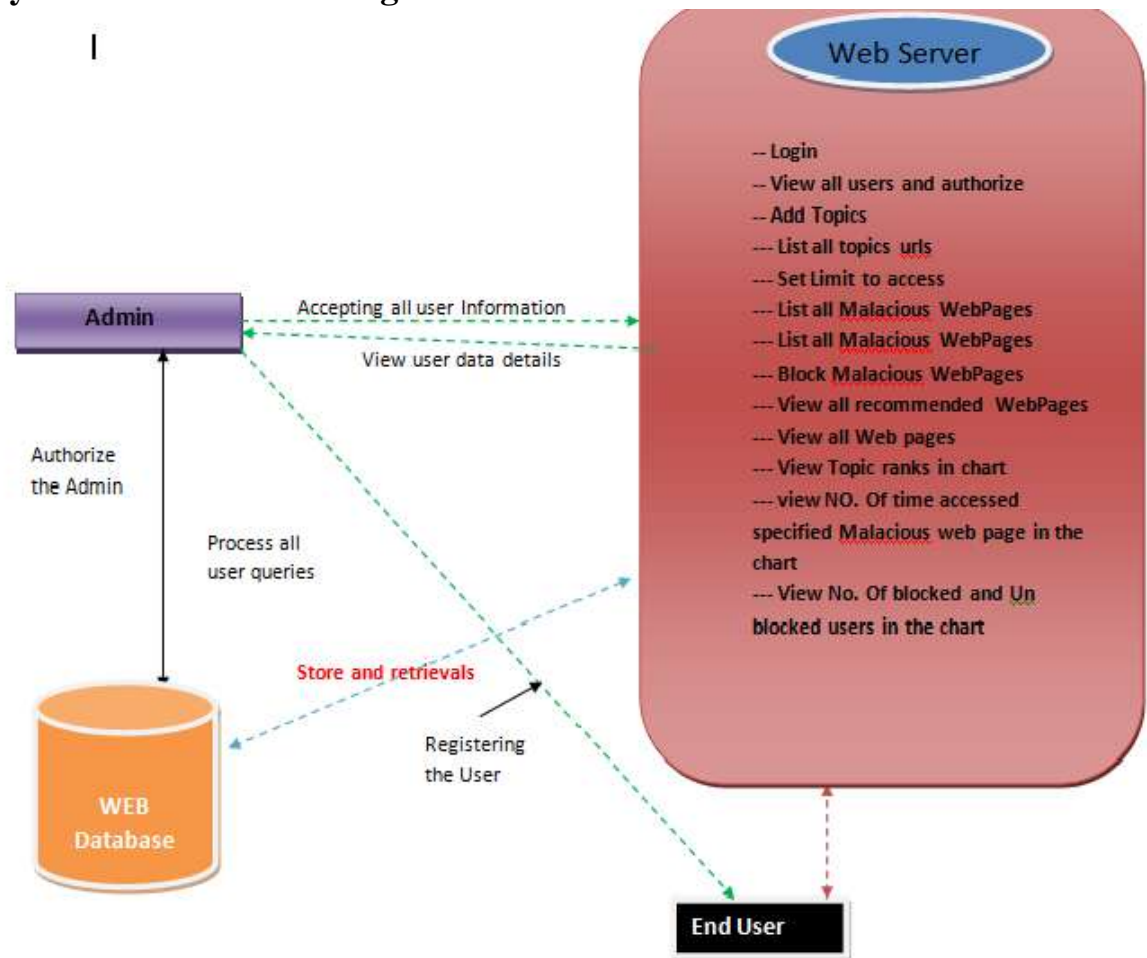


FIG 5.1 : system architecture

- Register and Login-- If he is blocked then show "YOU ARE BLOCKED"
- View profile
- Search WebPages by content keyword and Find malicious websites
- View all other user recommended Web pages
- View Top k web pages ulrs

5.2.2 Flow Chart : Admin

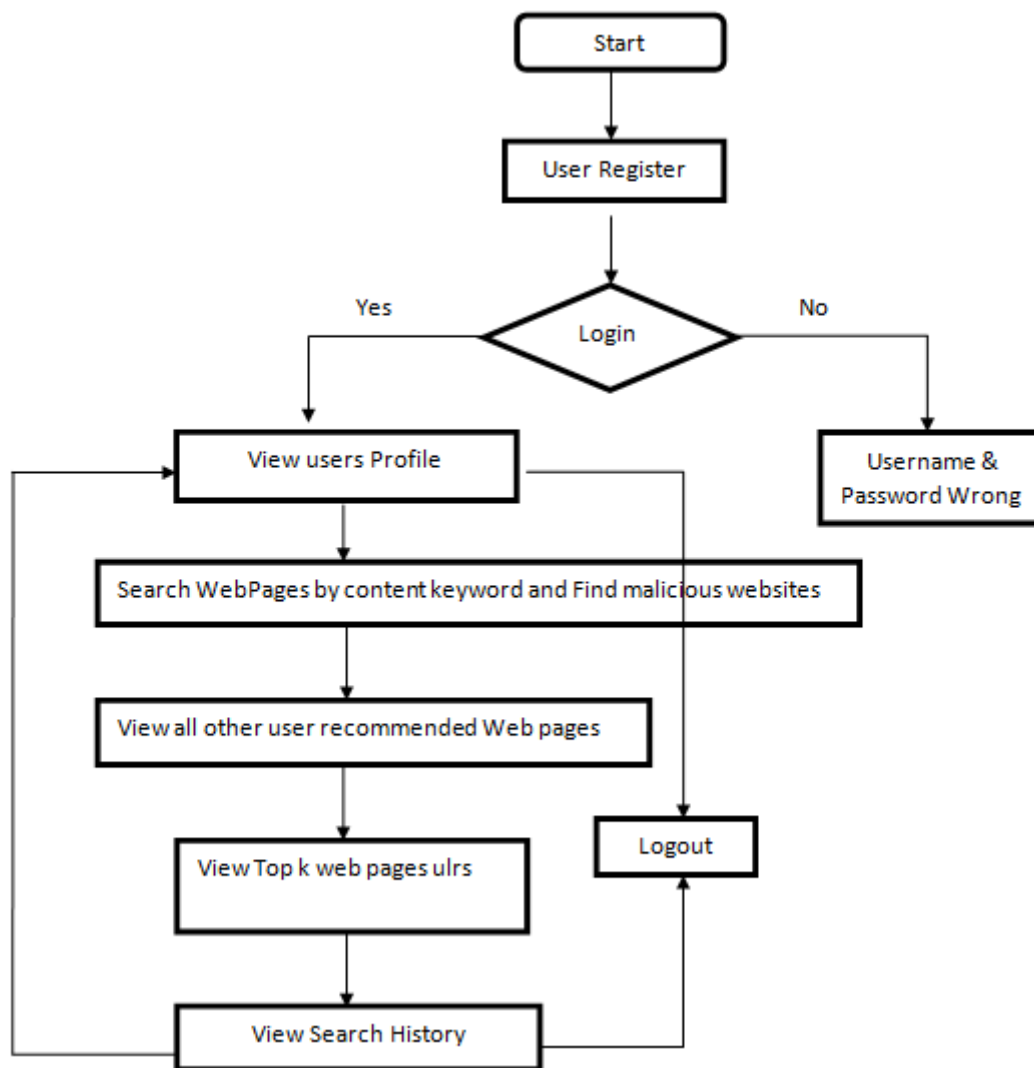


Fig 5.2: flow chart-admin

5.2.3 Flow Chart : user

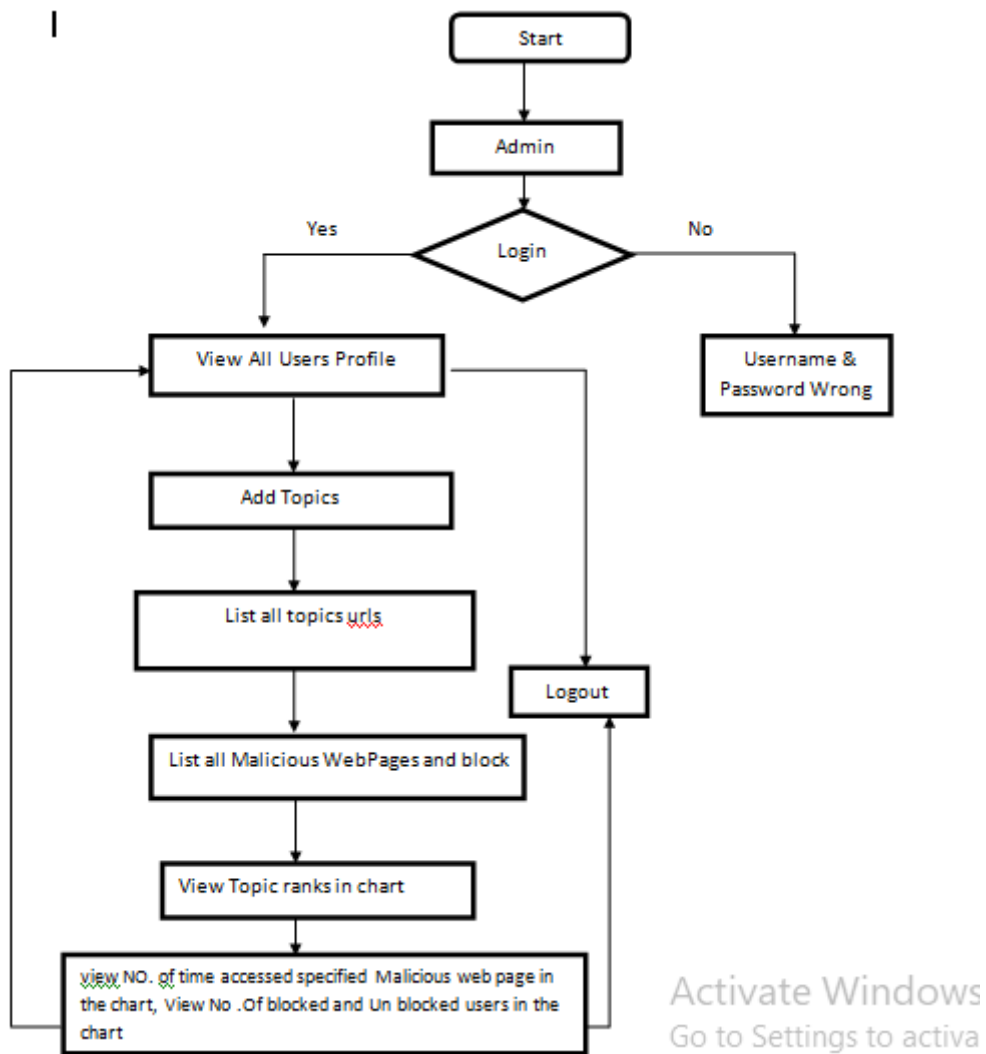


Fig 5.3: flow chart-admin

5.3 UML

5.3.1 HISTORY OF UML

Identifiable object-oriented modeling languages began to appear between mid-1970 and the late 1980s as various methodologies experimented with different approaches to object-oriented analysis and design. The number of identified modeling languages increased from less than 10 to more than 50 during the period between 1989-1994. Many users of OO method had trouble finding complete satisfaction in any one modeling language, fueling the “method wars”.

The development of UML began in late 1944 when Grady Brooch and Jim Rum Baugh of rational Software Corporation began their work on unifying the Brooch and his Objectors Company joined Rational and this unification effort, merging in the OOSE(Object-Oriented Software Engineering) method.

5.3.2 INTRODUCTION TO UML

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is very important part of developing object oriented software development process.

The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

5.3.3 DEFINITION

UML is a general purpose visual Unified Modeling Language that is used to specify, visualize, construct and document the artifacts of the software system.

- **UML as a Language**

It will provide vocabulary and rules for communication and function on conceptual and physical representation. So it is modeling language.

- **UML is a graphical language for**

- I. Visualization
- II. Constructing
- III. Documenting
- IV. Specifying

- **UML Visualization:** The UML includes both graphical and textual representation. It makes easy to visualize system and for better understanding.

- **UML Constructing:**UML models can be directly connected to a variety of programming languages and it's sufficiently expressive and free from any ambiguity to permit the direct execution of models.
- **UML Documenting:**UML provides variety of documents in addition raw executable codes.
- **UML Specifying:**Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decision that must be made in developing and displaying a software intensive system.

5.3.4 UML DIAGRAMS:

- **Unified Modeling Language:**The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the systems from distinctly with different perspective. Each view is defined by a set of diagram, which is as follows.
- **User Model View:**This view represents the system from the user's perspective.The analysis representation describes a usage scenario from the end-users perspective.
- **Structural Model View:**In this model the data and functionality are arrived from inside the system.This model view models the static structures.
- **Behavioral Model View:**It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
- **Implementation Model View:**In this the structural and behavioral as parts of the system are represented as they are to be built.
- **Environmental Model View:**In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

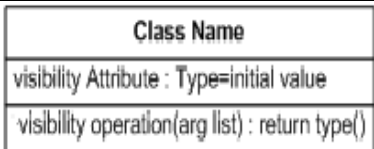



UML is specifically constructed through two different domains they are:

- i. UML Analysis modeling, this focuses on the user model and structural model views of the system.
- ii. UML design modeling which focuses on the behavioral modeling, implementation modeling and environmental model views.


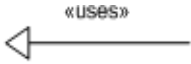


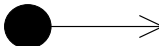
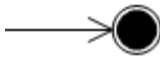

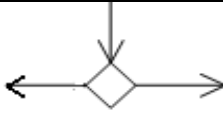
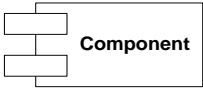
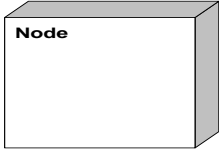
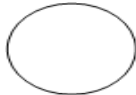
These diagrams are divided into 5 types,




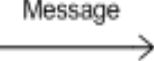
- **Use Case Diagram:** A use case diagram shows a set of use cases and actors and their relationships. Use case diagram address the static use case view of the system.
- **Class Diagram:** A class diagram shows a set of classes, interfaces, and collaborations and their relationship. These diagrams are not most common diagram found in modelling object-oriented system.
- **Sequence Diagram:** An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
- **Activity Diagram:** It illustrates the dynamic nature of the system by modelling the flow of control from activity to activity.
- **Collaboration Diagram:** It will model the interactions between objects in sequence they describe both dynamic and static behavior structure of system

5.4 LIST OF SYMBOLS:

S.No	SYMBOL NAME	SYMBOL	DESCRIPTION
1	Class		Classes represent a collection of similar entities grouped together.
2	Association		Association represents a static relationship between classes.
3	Aggregation		Aggregation is a form of association. It aggregates several classes into single class.
4	Actor		Actors are the users of the system and other external entity that react with the system.

DETECTING HARMFUL MOBILE WEBPAGES

5	Use Case		A use case is a interaction between the system and the external environment.
6	Relation (Uses)		It is used for additional process communication.
7	Communication		It is the communication between various use cases.
8	State		It represents the state of a process. Each state goes through various flows.
9	Initial State		It represents the initial state of the object.
10	Final State		It represents the final state of the object.
11	Control Flow		It represents the various control flow between the states.
12	Decision Box		It represents the decision making process from a constraint.
13	Component		Components represent the physical components used in the system.
14	Node		Deployment diagrams use the nodes for representing physical modules, which is a collection of components.
15	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.

16	External Entity		It represent any external entity such as keyboard, sensors etc which are used in the system.
17	Transition		It represent any communication that occurs between the processes.
18	Object Lifeline		Object lifelines represents the vertical dimension that objects communicates.
19	Message		It represents the messages exchanged.

5.5 UML DIAGRAMS

5.5.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the case and will often be accompanied by other types of diagrams as well.

A use case diagrams shows a set of use cases and actors and their relationships. Use case diagrams are especially important in organizing and modelling the behavior of the system.

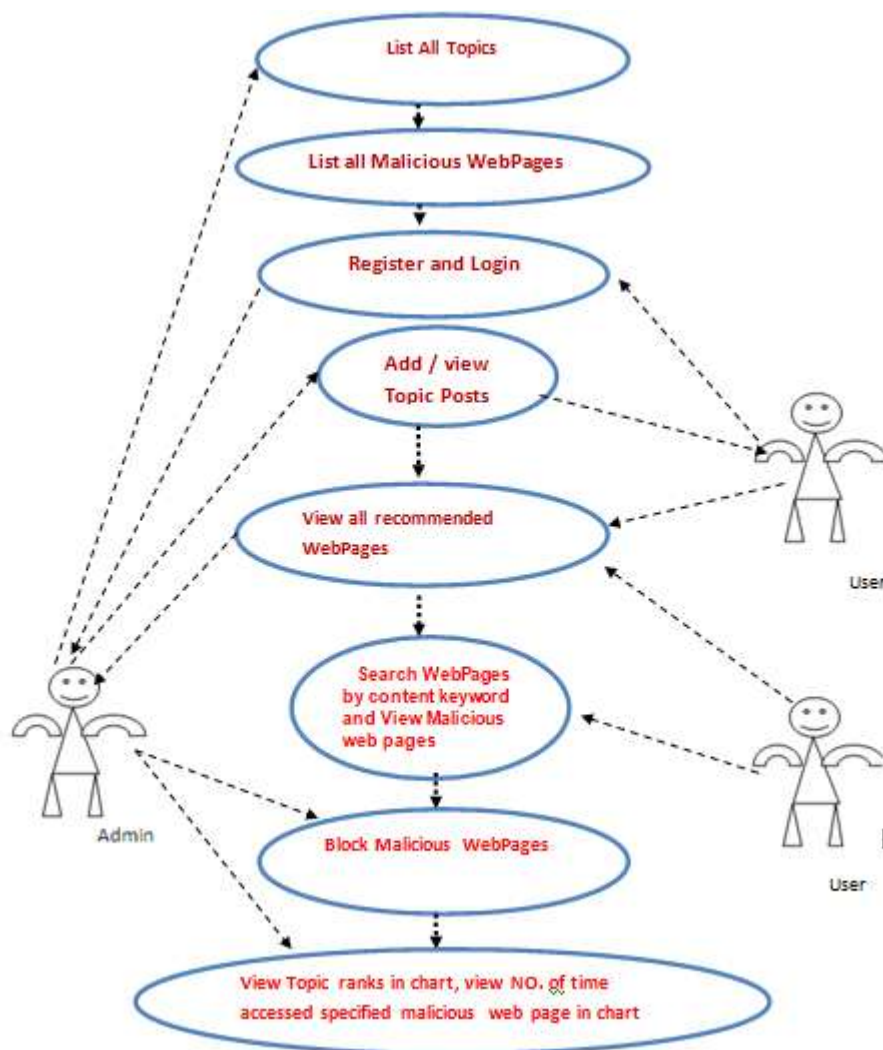


Fig 5.4 Use Case Diagram

5.5.2 CLASS DIAGRAM

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematics of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

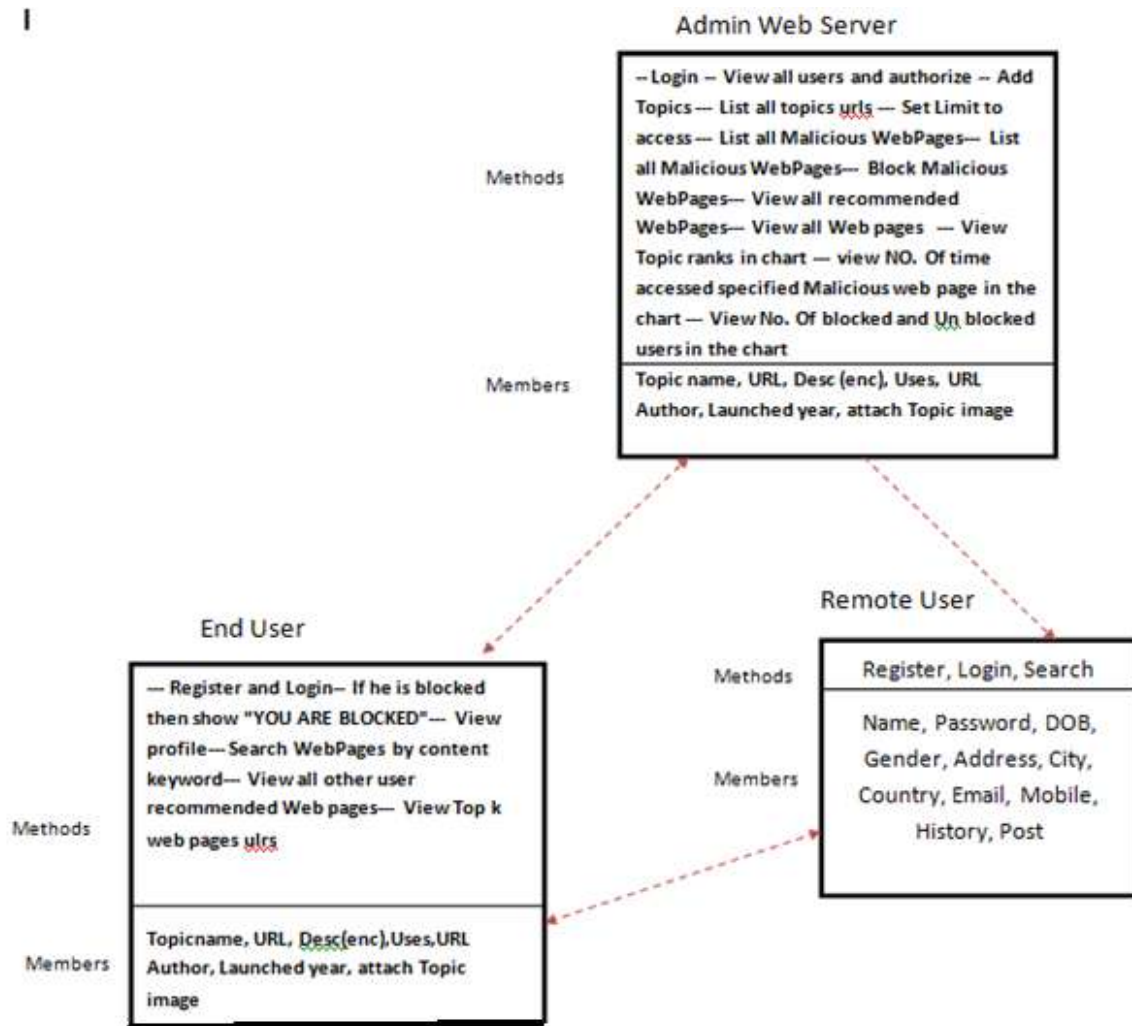


Fig 5.5 Class Diagram

5.5.3 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

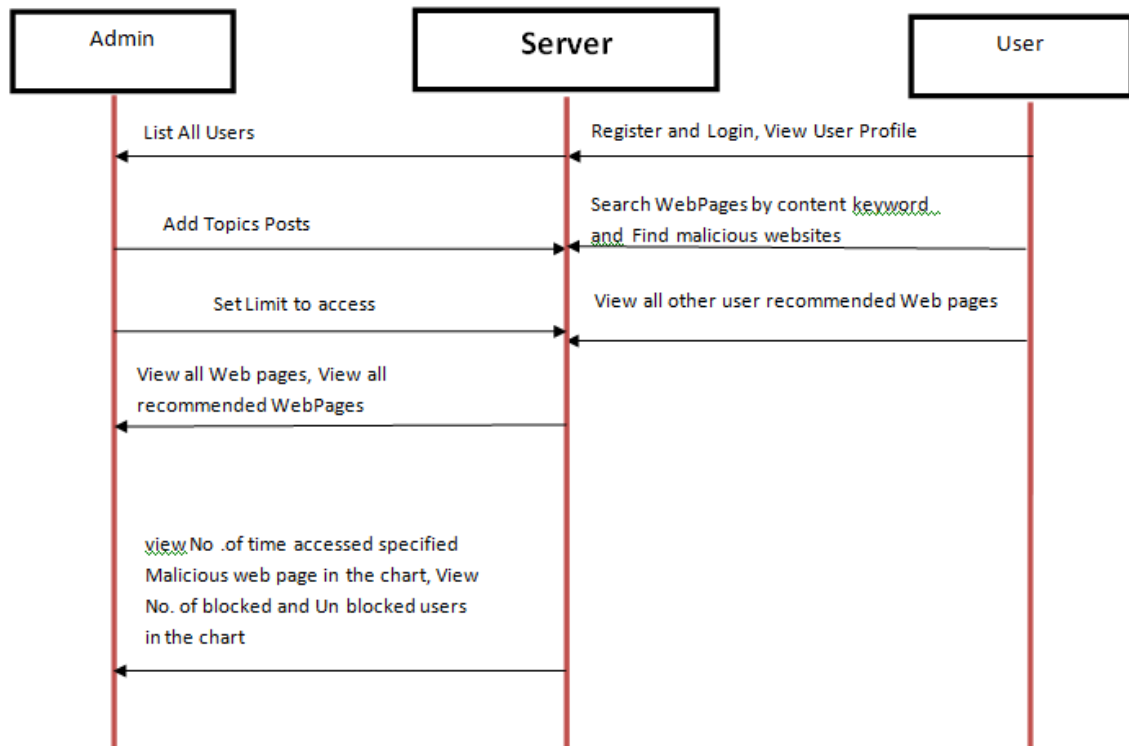


Fig 5.6 Sequence Diagram

5.5.4 DATA FLOW DIAGRAM

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

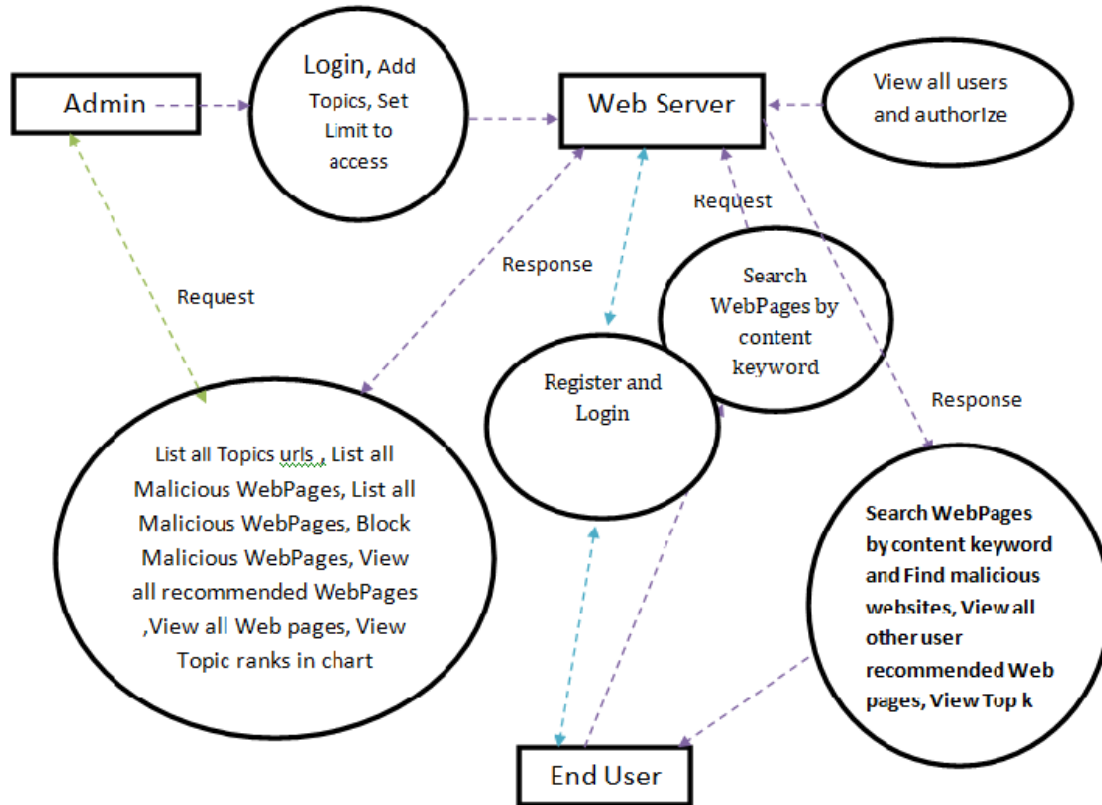


Fig 5.7 Sequence Diagram

5.6 DATABASE DESIGN

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The MySQL database has been chosen for developing the relevant databases.

The following are the tables that are involved in the proposed system.

5.6.1 Admin, user Login

Field Name	Data Type	Description
User id	Varchar	User name
Password	Varchar	Password of the dean

Table 5.1 *Login Details*

5.6.2 Student Data

Field name	Data type	Description
Regno	Varchar	Regd number of the student
Marks	Int	Marks of the Student

Table 5.2 *Student Details*

6. IMPLEMENTATION

6.1 TECHNICAL DESCRIPTION

6.1.1 HYPER TEXT MARKUP LANGUAGE (HTML)

Hypertext Markup Language (HTML), the languages of the World Wide Web (www), allows users to produce Web pages that include text, graphics and pointer to other Webpages (Hyperlinks). HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference.

A Markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, colour, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

6.1.2 CASCADING STYLE SHEETS (CSS)

CSS is used to style and lay out web pages — for example, to alter the font, color, size and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module gets you started on the path to CSS mastery with the basics of how it works, including selectors and properties, writing CSS rules, applying CSS to HTML, how to specify length, color, and other units in CSS, cascade and inheritance, and debugging CSS.

CSS (Cascading Style Sheets) allows you to create great looking web pages, but how does it work under the hood? This article explains what CSS is, how the browser turns HTML into a Document Object Model (DOM), how CSS is applied to parts of the DOM, some very basic syntax examples, and what code is used to actually include our CSS in our web page.

6.1.3 INTRODUCTION TO JAVA PROGRAMMING

Java is a simple and yet powerful object oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991. It was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. It was developed to provide a platform-independent programming language.

- **Platform Independent:**

Unlike many other programming languages including C and C++ when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

- **Java Virtual Machine:**

What is the Java Virtual Machine? What is its role?

Java was designed with a concept of ‘write once and run everywhere’. Java Virtual Machine plays the central role in this concept. The JVM is the environment in which Java programs execute. It is software that is implemented on top of real hardware and operating system. When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files. The JVM executes these byte codes. So Java byte codes can be thought of as the machine language of the JVM. A JVM can either interpret the byte code one instruction at a time or the byte code can be compiled further for the real microprocessor using what is called a just-in-time compiler. The JVM must be implemented on a particular platform before compiled programs can run on that platform.

- **Object Oriented Programming:**

Object Oriented Programming is a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of a class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

- **OOP Concepts**

Four principles of Object Oriented Programming are:

- I. Abstraction
- II. Encapsulation
- III. Inheritance
- IV. Polymorphism

I. Abstraction:

Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

II. Encapsulation:

Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

- Hides the implementation details of a class.
- Forces the user to use an interface to access data
- Makes the code more maintainable.

III. Inheritance:

Inheritance is the process by which one object acquires the properties of another object.

IV. Polymorphism:

Polymorphism is the existence of the classes or methods in different forms or single name denoting different implementations.

- **Java has powerful features. The following are some of them:**

Since Java is an object oriented programming language it has following features:

- Reusability of Code
- Emphasis on data rather than procedure
- Data is hidden and cannot be accessed by external functions
- Objects can communicate with each other through functions
- New data and functions can be easily added
- Simple
- Reusability of Code
- Portable (Platform Independent)
- Distributed
- Robust
- Secure
- High Performance
- Dynamic
- Threaded

- Interpreted

- **Java is distributed:**

With extensive set of routines to handle TCP/IP protocols like HTTP and FTP java can open and access the objects across net via URLs.

- **Java is multi-threaded:**

One of the powerful aspects of the Java language is that it allows multiple threads of execution to run concurrently within the same program A single Java program can have many different threads executing independently and continuously. Multiple Java applets can run on the browser at the same time sharing the CPU time.

- **Java is secure:**

Java was designed to allow secure execution of code across network. To make Java secure many of the features of C and C++ were eliminated. Java does not use Pointers. Java programs cannot access arbitrary addresses in memory.

- **Garbage Collection:**

Automatic garbage collection is another great feature of Java with which it prevents inadvertent corruption of memory. Similar to C++, Java has a new operator to allocate memory on the heap for a new object. But it does not use delete operator to free the memory as it is done in C++ to free the memory if the object is no longer needed. It is done automatically with garbage collector.

- **Java Applications:**

Java has evolved from a simple language providing interactive dynamic content for web pages to a predominant enterprise-enabled programming language suitable for developing significant and critical applications. Today, It is used for many types of applications including Web based applications, Financial applications, Gaming applications, embedded systems, Distributed enterprise applications, mobile applications, Image processors, desktop applications and many more.

6.1.4 JAVA SERVER PAGE (JSP)

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified. JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. This is a type of Model 2 architecture.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

JSPs are usually used to deliver HTML and XML documents, but through the use of `OutputStream`, they can deliver other types of data as well.

The Web container creates JSP implicit objects like request, response, session, application, config, page, pageContext, out and exception. JSP Engine creates these objects during translation phase.

6.1.5 JAVA DATABASE CONNECTIVITY (JDBC)

Sun Microsystems released JDBC as part of Java Development Kit (JDK) 1.1 on February 19, 1997. Since then it has been part of the Java Platform, Standard Edition (Java SE). The JDBC classes are contained in the Java package `java.sql` and `javax.sql`.

Starting with version 3.1, JDBC has been developed under the Java Community Process. JSR 54 specifies JDBC 3.0 (included in J2SE 1.4), JSR 114 specifies the JDBC Rowset additions, and JSR 221 is the specification of JDBC 4.0 (included in Java SE 6).

JDBC 4.1, is specified by a maintenance release 1 of JSR 221 and is included in Java SE 7.

JDBC 4.2, is specified by a maintenance release 2 of JSR 221 and is included in Java SE 8.

The latest version, JDBC 4.3, is specified by a maintenance release 3 of JSR 221 and is included in Java SE 9.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

Statement – the statement is sent to the database server each and every time.

PreparedStatement – the statement is cached and then the execution path is the pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

CallableStatement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information. Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types. There is an extension to the basic JDBC API in the javax.sql.

6.1.6 JAVA VIRTUAL MACHINE (JVM)

The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the blue book, The preface states: We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

One of Oracle's JVMs is named HotSpot, the other, inherited from BEA Systems is JRockit. Clean-room Java implementations include Kaffe and IBM J9. Oracle owns the Java trademark and may allow its use to certify implementation suites as fully compatible with Oracle's specification.

6.1.7 JAVA DEVELOPMENT KIT (JDK)

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

People new to Java may be confused about whether to use the JRE or the JDK. To run Java applications and applets, simply download the JRE. However, to develop Java applications and applets as well as run them, the JDK is needed. Java developers are initially presented with two JDK tools, java and javac. Both are run from the command prompt. Java source files are simple text files saved with an extension of .java. After writing and saving Java source code, the javac compiler is invoked to create .class files. Once the .class files are created, the 'java' command can be used to run the java program. For developers who wish to work in an integrated development environment (IDE), a JDK bundled with Netbeans can be downloaded from the Oracle website. Such IDEs speed up the development process by introducing point-and-click and drag-and-drop features for creating an application.

There are different JDKs for various platforms. The supported platforms include Windows, Linux and Solaris. Mac users need a different software development kit, which includes adaptations of some tools found in the JDK.

The JDK forms an extended subset of a software development kit (SDK). It includes "tools for developing, debugging, and monitoring Java applications". Oracle strongly suggests to now use the term JDK to refer to the Java SE Development Kit. The Java EE SDK is available with or without the JDK, by which they specifically mean the Java SE 7 JDK.

- **How to set path in Java**

The path is required to be set for using tools such as javac, java etc.

If you are saving the java source file inside the jdk/bin directory, path is not required to be set because all the tools will be available in the current directory. But if you are having your java file outside the jdk/bin folder, it is necessary to set path of JDK.

There are 2 ways to set java path:

1. temporary

2. permanent

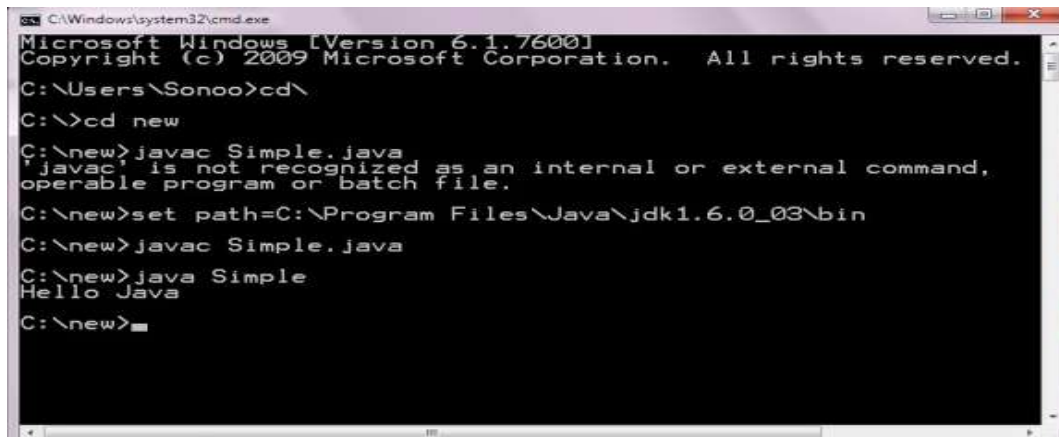
1. How to set Temporary Path of JDK in Windows

To set the temporary path of JDK, you need to follow following steps:

- Open command prompt
- copy the path of jdk/bin directory
- write in command prompt: set path=copied_path

For Example:

set path=C:\Program Files\Java\jdk1.6.0_23\bin



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Sonoo>cd\
C:\>cd new
C:\new>javac Simple.java
'javac' is not recognized as an internal or external command,
operable program or batch file.
C:\new>set path=C:\Program Files\Java\jdk1.6.0_03\bin
C:\new>javac Simple.java
C:\new>java Simple
Hello Java
C:\new>
```

2. How to set Permanent Path of JDK in Windows

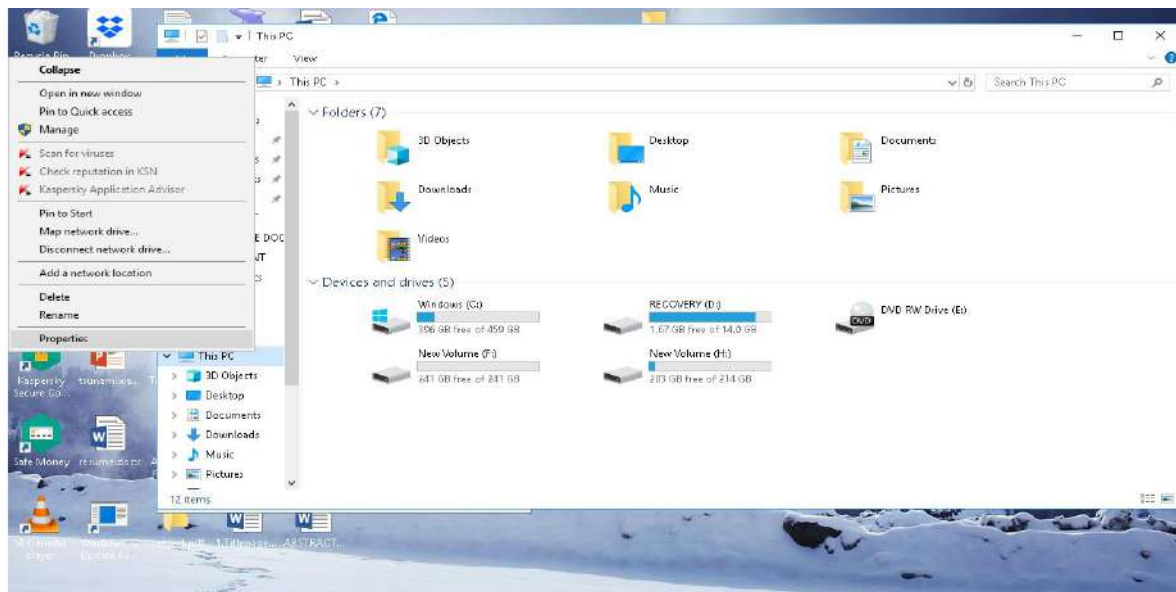
For setting the permanent path of JDK, you need to follow these steps:

- Go to My Computer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

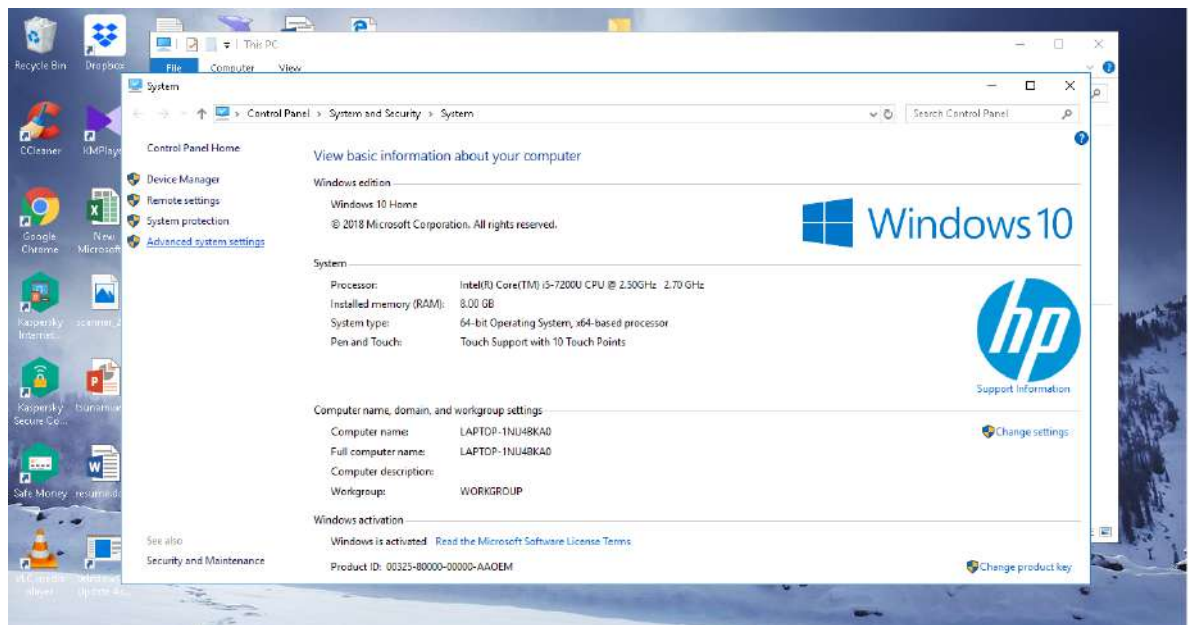
For Example:

- i. Go to My Computer properties

DETECTING HARMFUL MOBILE WEBPAGES

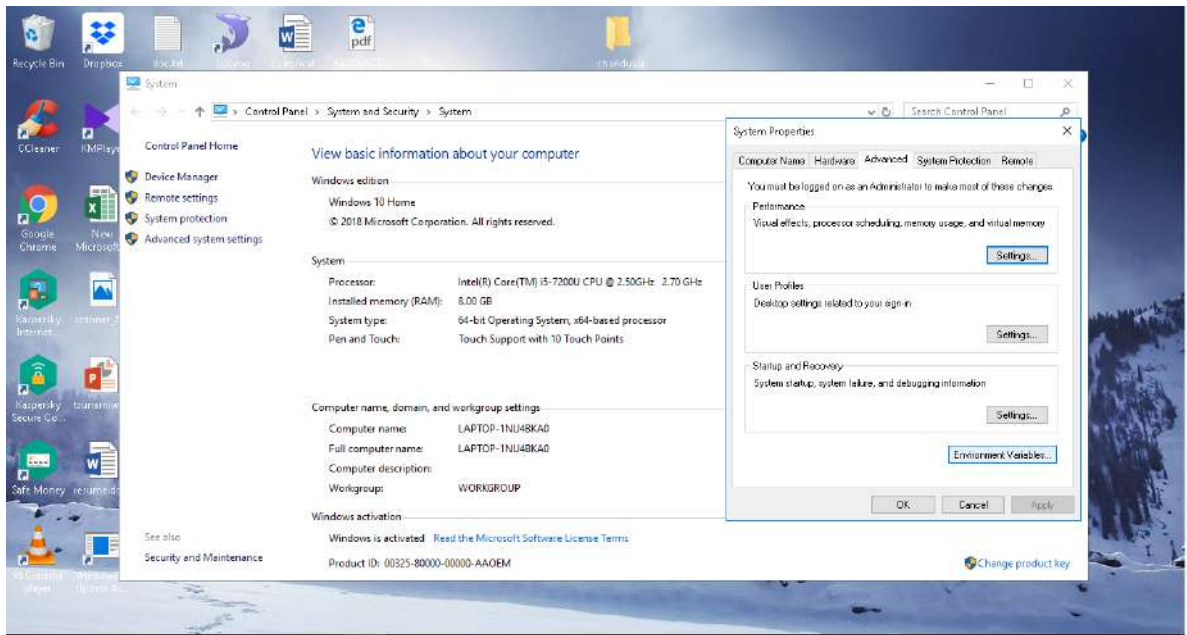


ii. Click on advanced tab

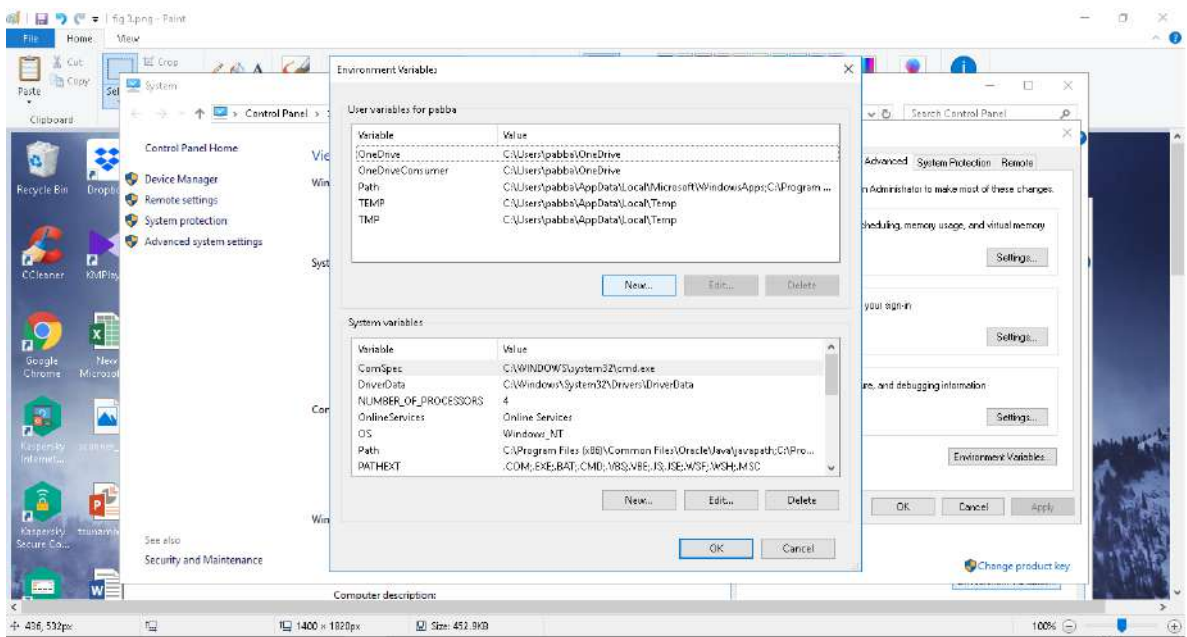


DETECTING HARMFUL MOBILE WEBPAGES

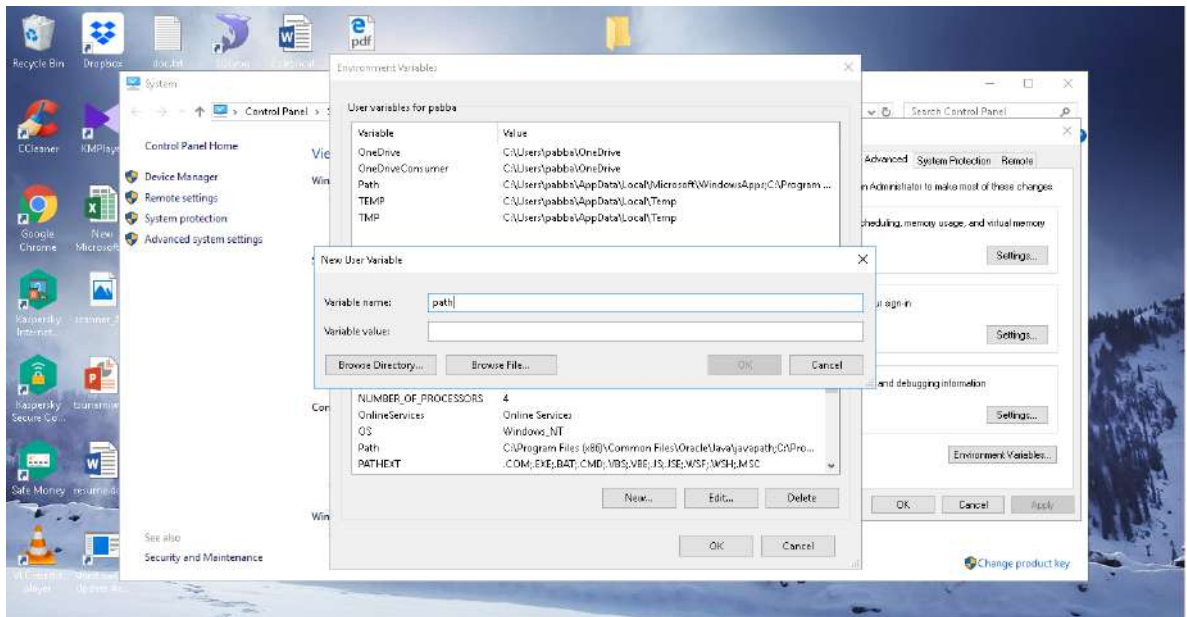
iii. Click on environment variables



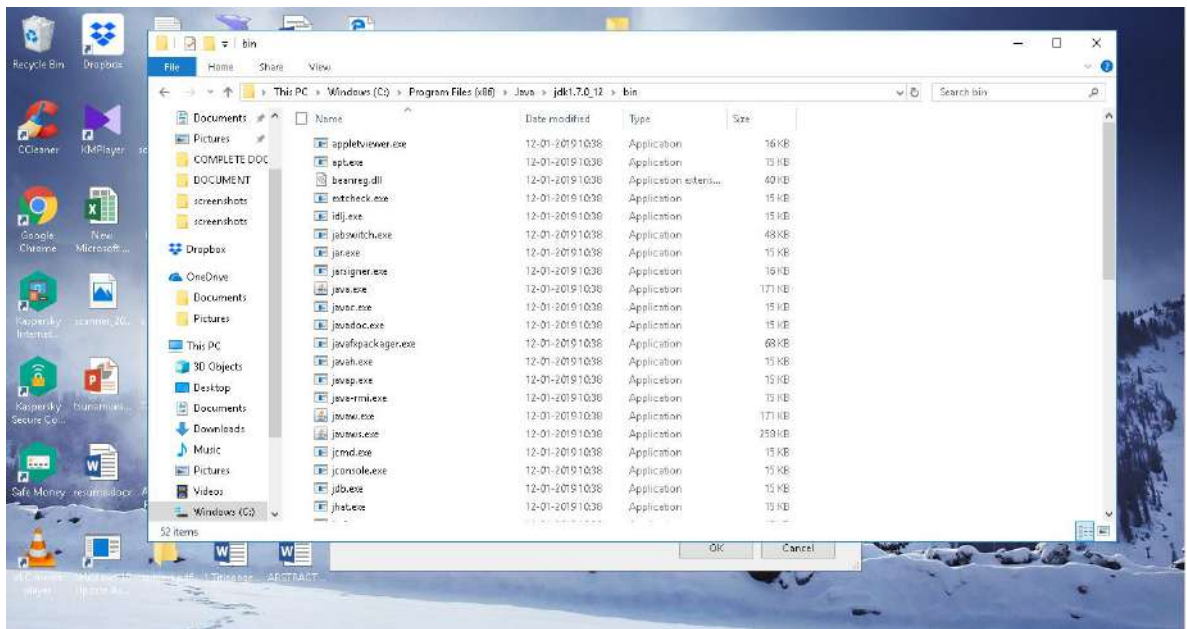
iv. click on new tab of user variables



- v. write path in variable name

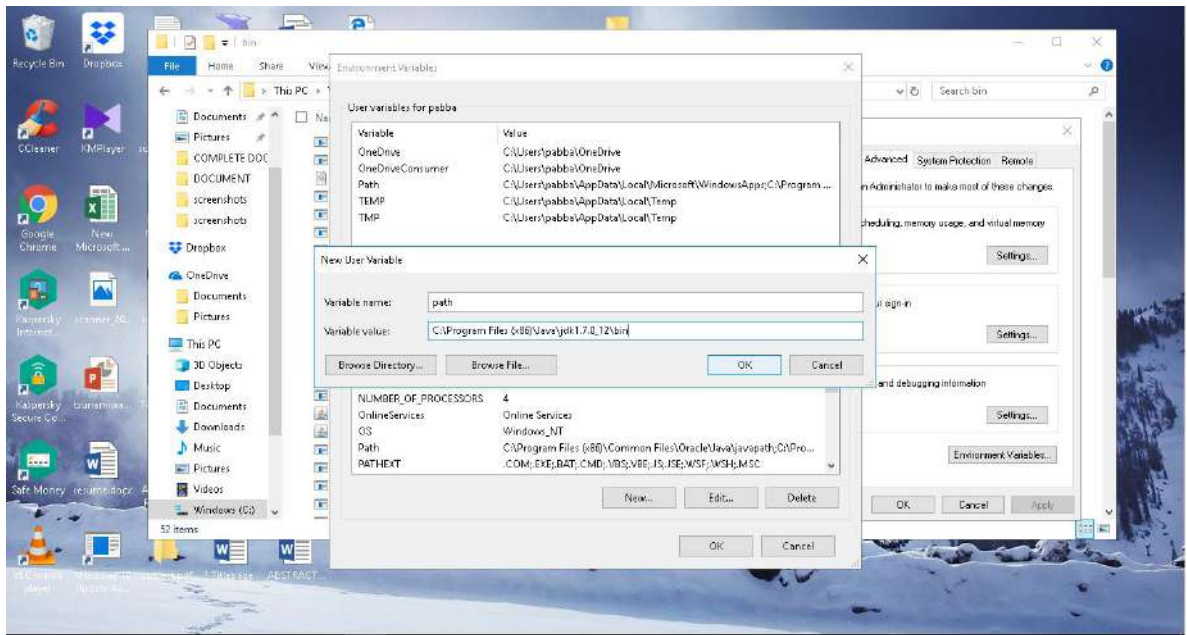


- vi. Copy the path of bin folder

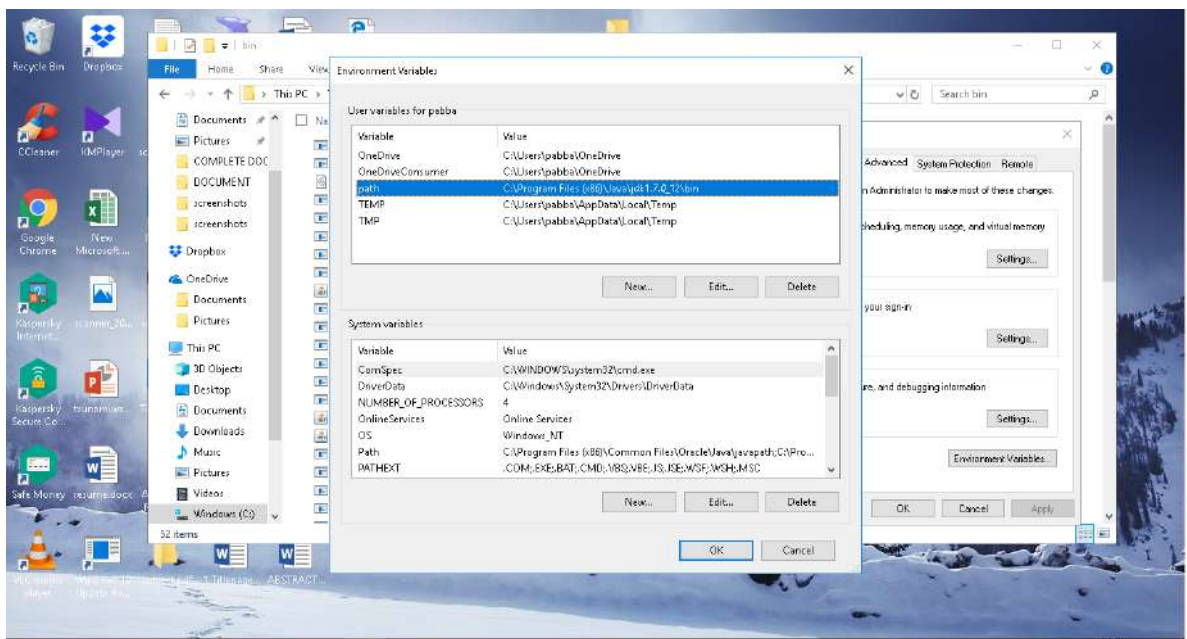


DETECTING HARMFUL MOBILE WEBPAGES

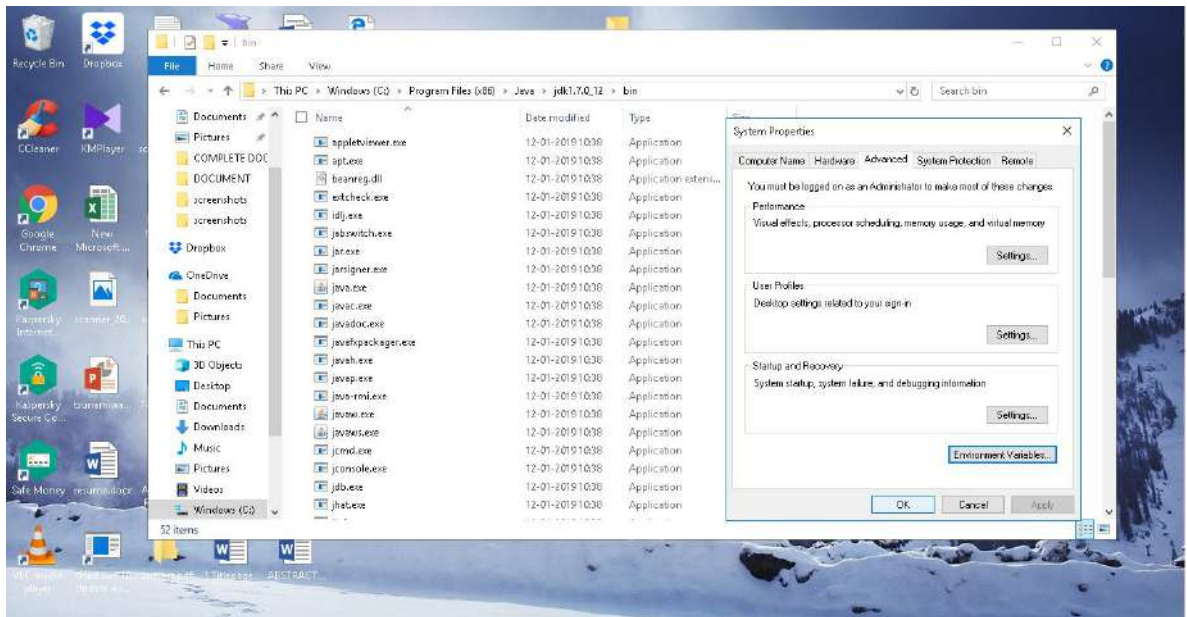
vii. paste path of bin folder in variable value



viii. click on ok button



- ix. click on ok button



Now your permanent path is set. You can now execute any program of java from any drive

6.1.8 MYSQL DATABASE

MySQL (officially pronounced as /maɪ ˈɛskjuːˈɛl/ "My S-Q-L") is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

CHAPTER-7

TESTING

7.1 TEST CASE DESCRIPTION

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

7.1.1 PSYCHOLOGY OF TESTING

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work. Testing is the process of executing a program with the intent of finding errors.

7.1.2 TESTING OBJECTIVES

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

7.2 LEVELS OF TESTING

In order to uncover the errors present in different phases we have the concept of levels of testing.

7.2.1 SYSTEM TESTING:

The philosophy behind testing is to find errors. Test cases are devised with this in mind. A strategy employed for system testing is code testing.

7.2.2 CODE TESTING:

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed on all systems.

7.3 TYPES OF TESTING

7.3.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

Giving different sets of inputs has tested each module. When developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

In this application developer tests the programs up as system. Software units in a system are the modules and routines that are assembled and integrated to form a specific function. Unit testing is first done on modules, independent of one another to locate errors. This enables to detect errors. Through this error resulting from interaction between modules initially avoided.

7.3.2 LINK TESTING

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

7.3.3 INTEGRATION TESTING

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

7.3.4 SYSTEM TESTING

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal's to see if software meets its requirements.

7.3.5 ACCEPTANCE TESTING

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development.

7.3.6 WHITE BOX TESTING

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

7.3.7 BLACK BOX TESTING

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

CHAPTER-8

SCREENSHOTS

8.1 Home Page



FIG 8.1 : Home Page

8.2 Admin Login Page



FIG 8.2 : Admin Login Page

8.3 Admin Home page

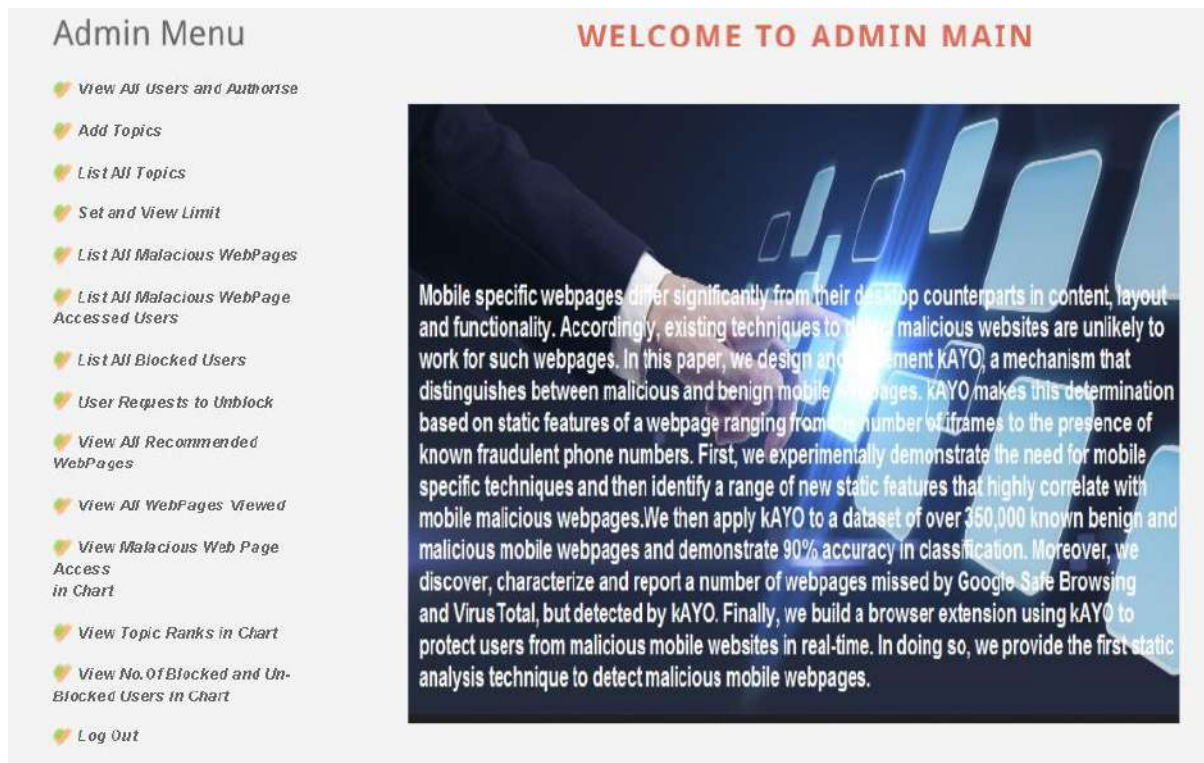


FIG 8.3 Admin Home page

8.4 View all users and authorise

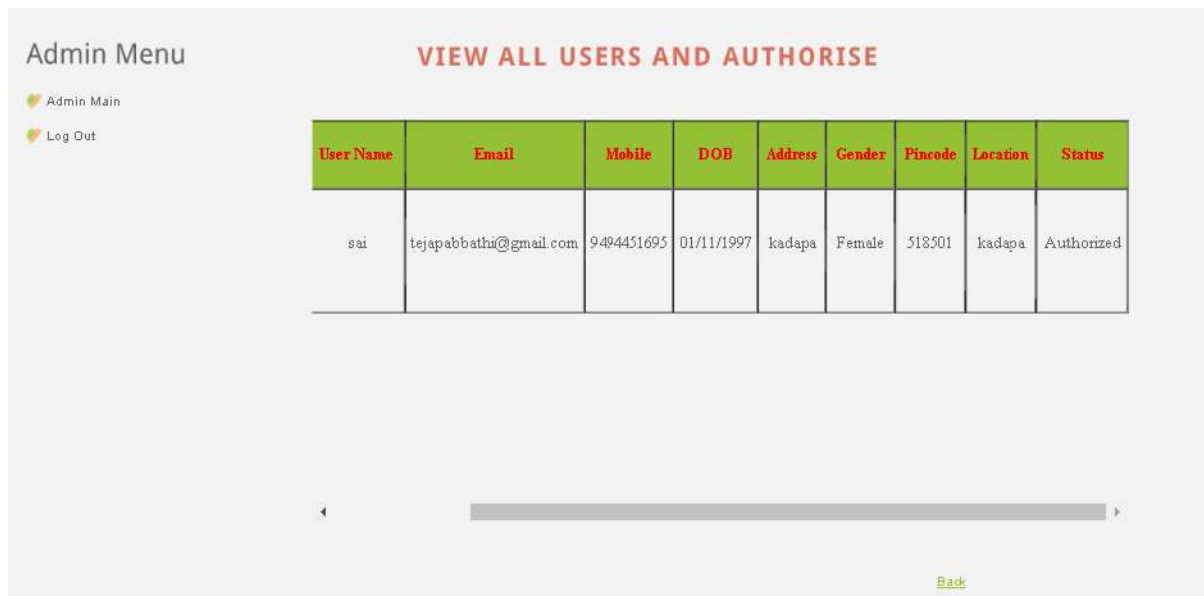


FIG 8.4 View all users and authorise

8.5 Add Topic

Admin Menu

- Admin Main
- Log Out

ADD TOPIC

Topic Name	hindu
Url	https://www.googleadservices.com/pagead/aclick?sa=L&ai=DChc
Description	news
Uses	news
Url Author	hindhu
Launched Year	2001
Attach Image	<input type="button" value="Choose File"/> No file chosen
<input type="button" value="Add"/> <input type="button" value="Reset"/>	

Back

FIG 8.5: Add Topic

8.6 View all Topics

Admin Menu

- Admin Main
- Log Out

View All TOPICS (Topics Based on Ranking)

Topic Name	Topic Image	URL	Description	Uses	Url Author	Launched Year
hindhu		http://localhost:8080/major/a_add_topic.jsp	dfasasf	asfd	dfa	232
sa		localhost:8080/major/attacker.jsp	sa	sa	sa	sa

FIG 8.6 View all Topics

8.7 Set limit to access malicious webpages

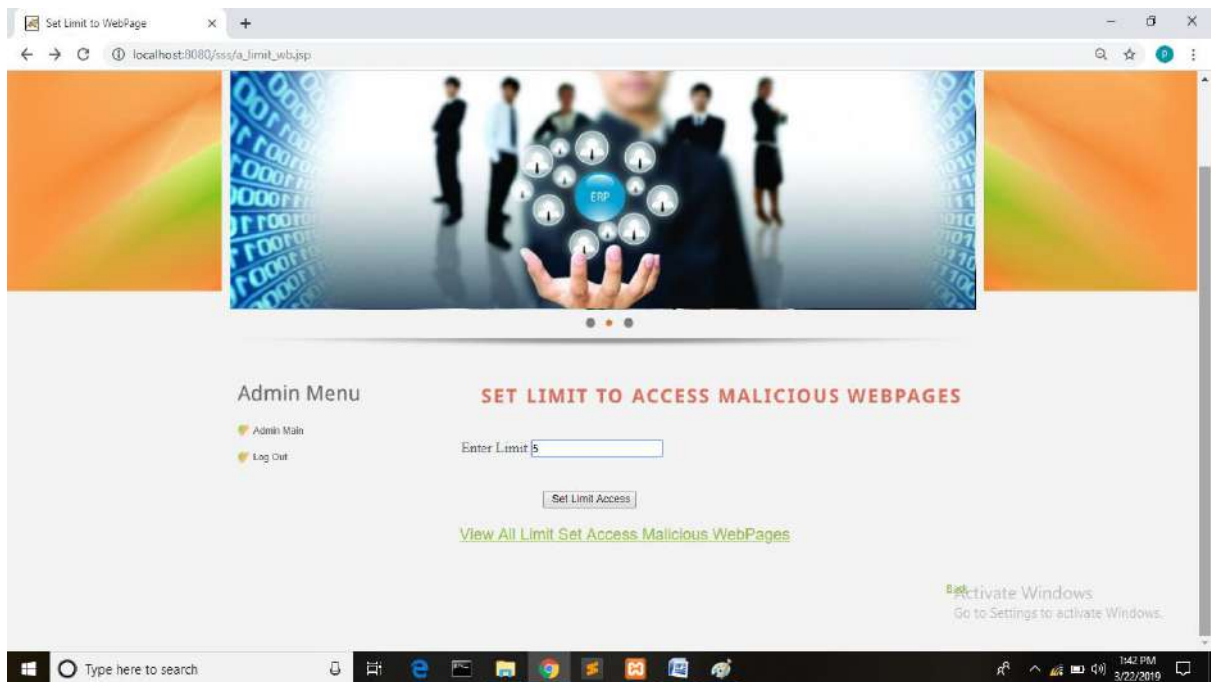


FIG 8.7 Set limit to access malicious webpages

8.8 View all malicious Webpages

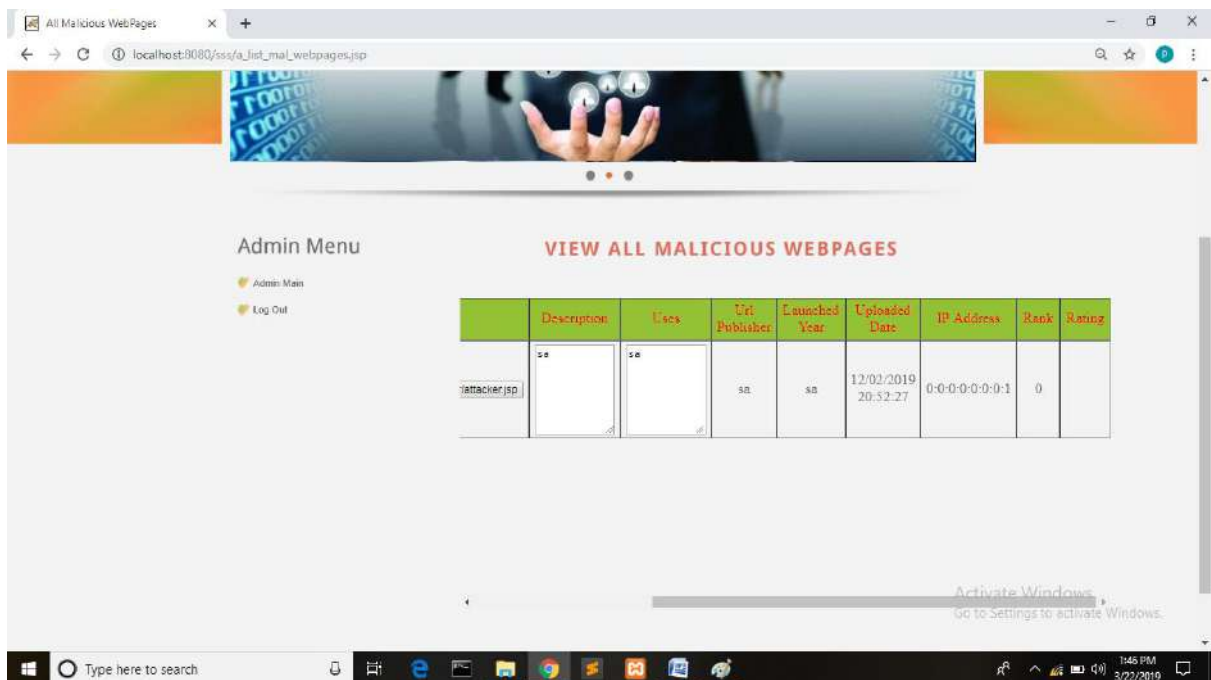


FIG 8.8 : View all malicious Webpages

8.9 View all webpages viewed

Admin Menu

- Admin Main
- Log Out

VIEW ALL WEBPAGES VIEWED

Id	User Name	Topic Name	Date	Ip Address
1	admin	hindhu	15/02/2019 10:34:56	0:0:0:0:0:0:1
2	admin	hindhu	15/02/2019 10:35:24	0:0:0:0:0:0:1
3	yamini	hindhu	19/02/2019 12:44:50	0:0:0:0:0:0:1
4	yamini	hindhu	20/02/2019 13:45:02	0:0:0:0:0:0:1

Activat
Go to Set
[Back](#)

FIG 8.9 : View all webpages viewed

8.10. user login page

Sidebar Menu

- Home Page
- Admin
- User
- Attacker

USER LOGIN



Name
Pasword

Login Reset

[New User? Register Here](#)

Activate W
Go to Setting

FIG 8.10 : user login page

8.11 User home page

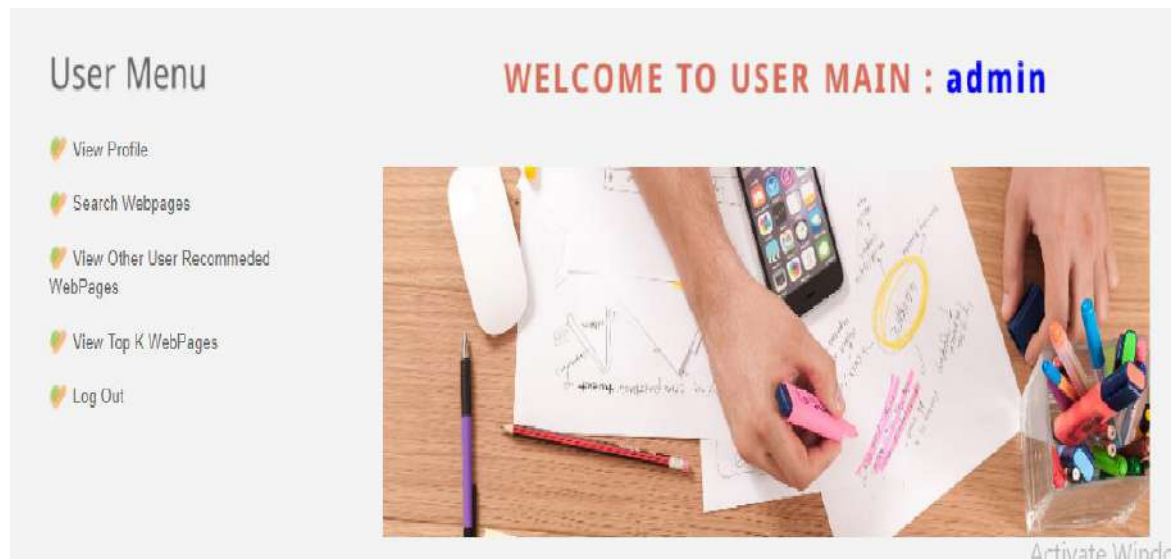


FIG 8.11 : User home page

8.12 user profile

User Menu

- User Main
- Log Out

MY PROFILE

	Name	admin
	E-Mail	THRUSHNA1998@GMAIL.COM
	Mobile	7989560717
	Location	nandyal
	Date of Birth	21-9-1974
	Address	25/81,SANJEEVA NAGAR
	Gender	Female
	Pincode	518501
	Status	Authorized

Activate Windows
Go to Settings to activate Windows.

FIG 8.12 : User Profile

8.13. Search webpages



FIG 8.13.: Search Webpage

8.14 search webpages result

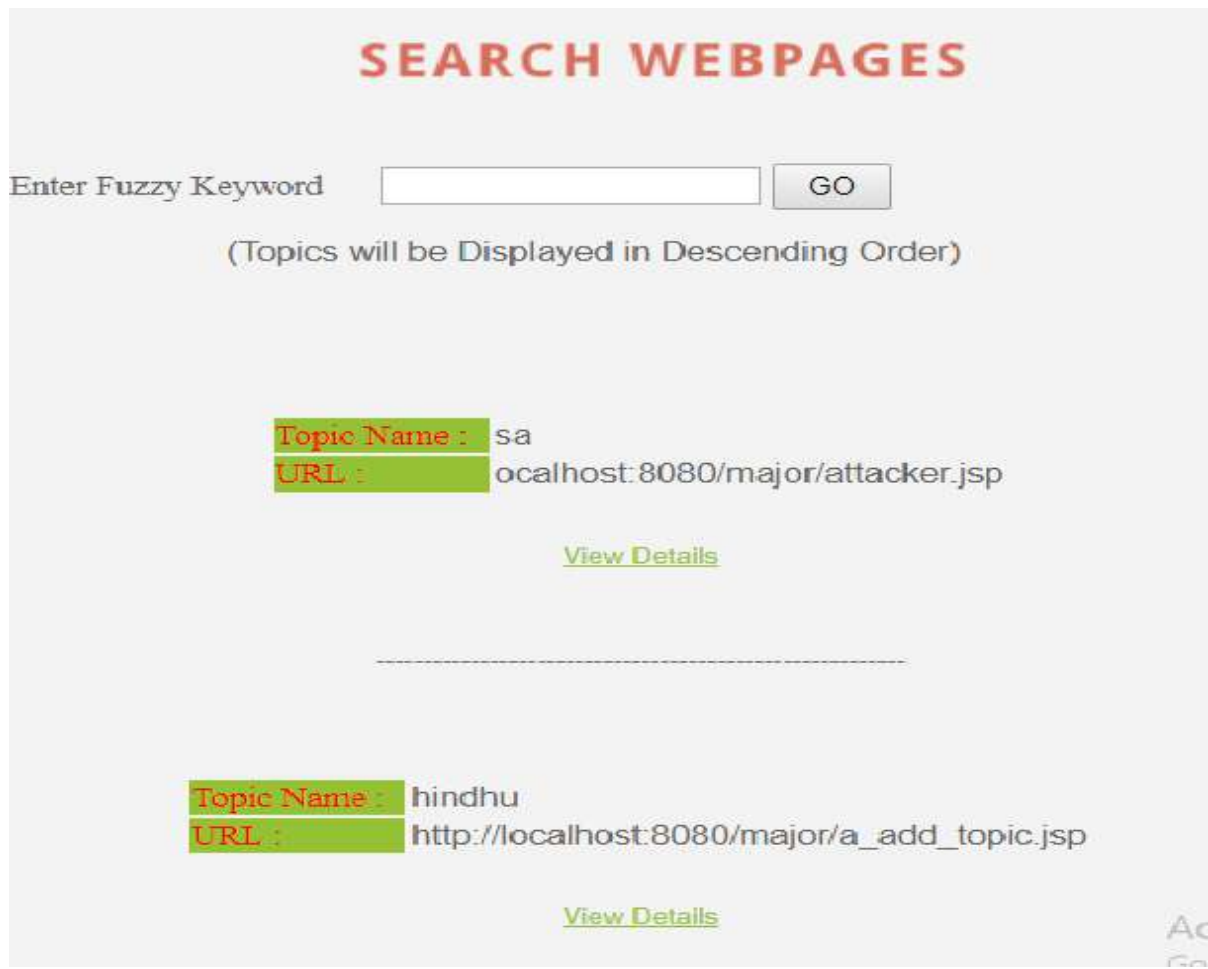


FIG 8.14 : Search webpages results

8.15 Blocked webpage



Fig 8.15 :Blocked webpages

8.16 webpage details

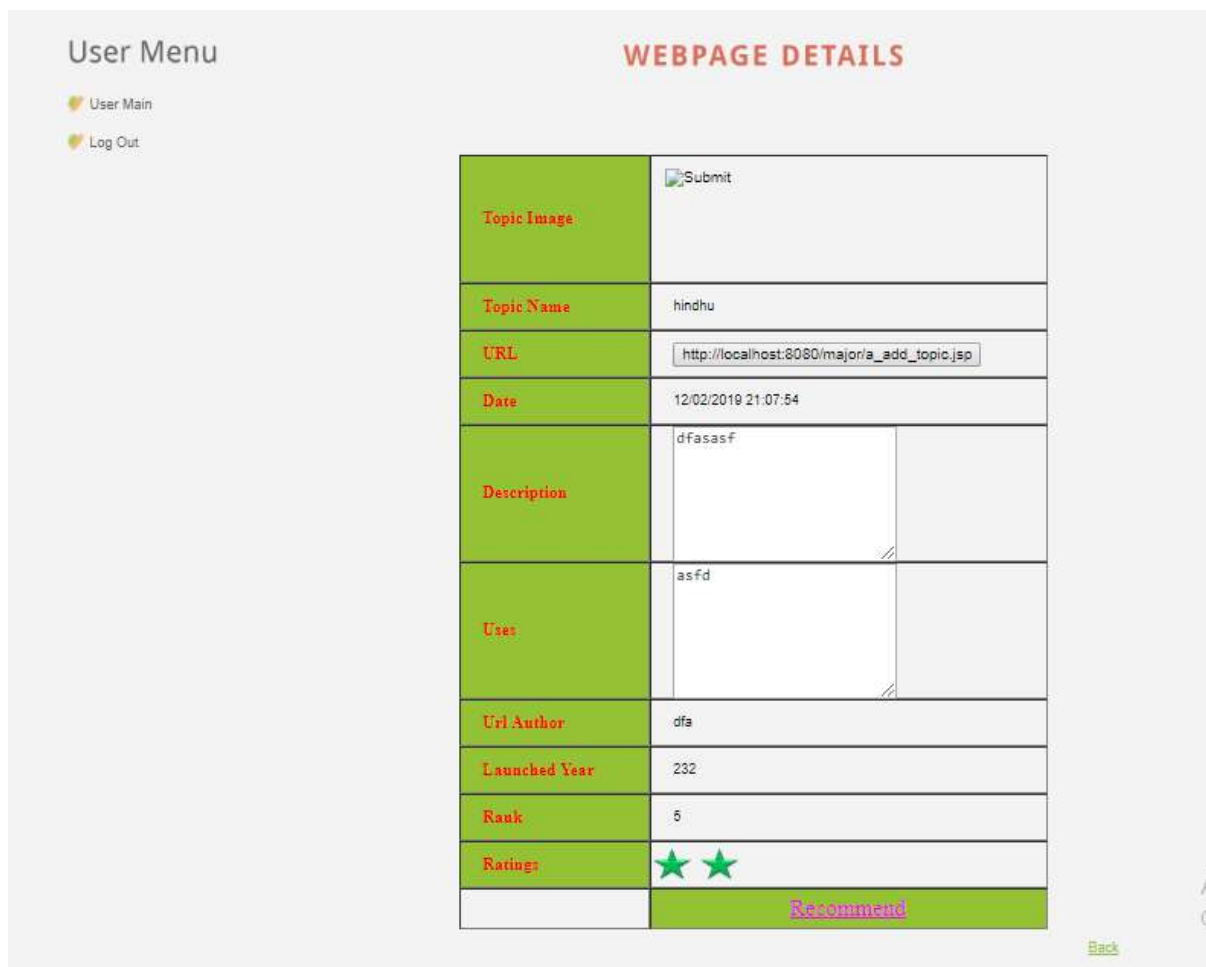


Fig 8.16:web page details

8.17 Attacker page

Sidebar Menu

- Home Page
- Admin
- User
- Attacker



ADD TOPIC

Topic Name	<input style="width: 90%;" type="text" value="hindhu"/>
Url	<input style="width: 90%;" type="text" value="http://localhost:8080/major/a_add_topic.jsp"/>
Description	<input style="width: 90%;" type="text" value="hindu"/>
Uses	<input style="width: 90%;" type="text" value="news"/>
Url Author	<input style="width: 90%;" type="text" value="dfa"/>
Launched Year	<input style="width: 90%;" type="text" value="232"/>
Attach Image	<input type="button" value="Choose File"/> No file chosen
<input type="button" value="Add"/> <input type="button" value="Reset"/>	

Activ
Go to

Fig 8.17 :Attacker Page

CHAPTER-9

CONCLUSION

Mobile webpages are significantly different than their desktop counterparts in content, functionality and layout. Therefore, existing techniques using static features of desktop webpages to detect malicious behavior do not work well for mobile specific pages. We designed and developed a fast and reliable static analysis technique called kAYO that detects mobile malicious webpages. kAYO makes these detections by measuring 44 mobile relevant features from webpages, out of which 11 are newly identified mobile specific features. kAYO provides 90% accuracy in classification, and detects a number of malicious mobile webpages in the wild that are not detected by existing techniques such as Google Safe Browsing and VirusTotal. Finally, we build a browser extension using kAYO that provides real-time feedback to users. We conclude that kAYO detects new mobile specific threats such as websites hosting known fraud numbers and takes the first step towards identifying new security challenges in the modern mobile web.

REFERENCES

- [1] Gnu octave: high-level interpreted language. <http://www.gnu.org/software/octave/>.
- [2] hphosts, a community managed hosts file. <http://hphosts.gt500.org/hosts.txt>.
- [3] Joewein.de LLC blacklist. <http://www.joewein.net/dl/bl/dom-bl-base.txt>.
- [4] Lookout. <https://play.google.com/store/apps/details?hl=en&id=com.lookout>.
- [5] Malware Domains List. <http://mirror1.malwaredomains.com/files/domains.txt>.
- [6] Phishtank. <http://www.phishtank.com/>.
- [7] Pindrop phone reputation service. <http://pindropsecurity.com/phone-fraud-solutions/phone-reputation-service-prs/>.
- [8] Scrapy — an open source web scraping framework for python. <http://scrapy.org/>.
- [9] VirusTotal. <https://www.virustotal.com/en/>.
- [10] Google developers: Safe Browsing API. <https://developers.google.com/safe-browsing/>, 2012.