AN INDUSTRY ORIENTED MINI PROJECT REPORT ON

SAFESURF-Chrome Extension

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

BACHELOR OF TECHNOLOGY

in

CSE (Cyber Security)

Submitted by

E SRIMANI TEJA 22B81A6251

T VIVEKANANDA 22B81A6263

K VIGNESHWAR 22B81A6260

Under the guidance of
G SAHITHI
Assistant Professor

DEPARTMENT OF CSE(Cyber Security)

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

APRIL 2025

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DEPARTMENT OF CSE (Cyber Security)

CERTIFICATE

This is to certify that the Industry Oriented Mini Project report entitled **"SAFESURF-Chrome Extension"** Bonafide record of work carried out by **E SRIMANI TEJA(22B81A6251), T VIVEKANANDA(22B81A6263)** and **K VIGNESHWAR(22B81A6260)** under the guidance of **Mrs. G. Sahithi, Assistant Professor** for the requirement of the award of **Bachelor of Technology** in **CSE (Cyber Security)** to the CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Project Guide

G. Sahithi
Assistant Professor
Department of CSE(CS)

Project Coordinator

Dr. C. Raghavendra
Associate Professor
Department of CSE(CS)

Head of the Department

External Examiner

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DECLARATION

We hereby declare that the Industry Oriented Mini Project report entitled **"SAFESURF-Chrome Extension"** is an original work done and submitted to CSE (Cyber Security) Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad in partial fulfilment for the requirement of the award of Bachelor of Technology in CSE (Cyber Security) and it is a record of bonafide project work carried out by us under the guidance of **G. SAHITHI,** Assistant Professor, Department of CSE (Cyber Security).

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree in this Institute or any other Institute or University.

Signature of the Student

E SRIMANI TEJA

Signature of the Student

T VIVEKANANDA

Signature of the Student

K VIGNESHWAR

Date:

Place:

ACKNOWLEDGEMENT

We are thankful for and fortunate enough to get constant encouragement, support, and guidance from all **Teaching staff of CSE (Cyber Security) Department** which helped us in successfully completing this Industry Oriented Mini Project.

We are extremely thankful to our internal guide **G. Sahithi,** Assistant Professor, Department of CSE (Cyber Security) for providing support and guidance, which made us complete the Industry Oriented Mini Projecton time.

We thank **Dr. C. Raghavendra**, Project Coordinator and **Mrs. G. Sahithi, Mrs. A. Geetha,** Project Review Committee members for their valuable guidance and support which helped us to complete the Industry Oriented Mini ProjectWork successfully.

We thank Professor and Head of the Department **Dr**. **M. Sunitha,** for giving us all the support and guidance which made us to complete the Industry Oriented Mini Project duly.

We would like to express heartful thanks to **Dr. H. N. Lakshmi,** Associate Dean, ET Department, for providing us an opportunity and extending support and guidance.

We thank our Vice-Principal **Prof. L. C. Siva Reddy** for providing excellent computing facilities and a disciplined atmosphere for doing our work.

We wish a deep sense of gratitude and heartfelt thanks to our Principal **Dr. K. Rama Mohan Reddy** and the **Management** for providing excellent lab facilities and tools.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In an era where cyber threats are becoming increasingly sophisticated, ensuring a safe browsing experience is paramount. **SAFESURF** is a lightweight Chrome extension developed using **JS** and the **VIRUS-TOTAL API** to enhance online security by providing real-time monitoring and scanning of URLs and downloaded files. Unlike traditional security tools that focus solely on either URL analysis or file scanning, SAFESURF integrates both functionalities to offer **comprehensive protection against phishing, malware, and other cyber threats**.

The extension leverages threat intelligence APIs, including **VirusTotal** and **Google Safe Browsing**, to analyze URLs and file signatures in real time. It employs browser APIs such as **webRequest** for network request interception and **downloads** for file monitoring, ensuring that users are safeguarded against malicious content before exposure. The HTML & JS based user interface provides an intuitive dashboard where users can **view scan results, receive threat notifications, and manage security settings**, including **customizable whitelisting** of trusted URLs and blacklisting of untrusted URLs.

By combining robust backend security mechanisms with a user-friendly interface, SAFESURF aims to bridge the gap in existing browser security tools. It empowers users with **proactive threat detection, real-time alerts, and customizable security controls**, ensuring a **safer and more secure** browsing experience.

CHAPTER 1
INTRODUCTION

1.1 Motivation

With cyber threats like phishing and malware increasing rapidly, many users unknowingly visit harmful websites or download malicious files that can compromise their systems. Existing security tools typically focus on either URL scanning or file scanning, but not both. This leaves a security gap, allowing cyber attackers to exploit the weakest point. SafeSurf was developed to fill this gap by integrating real-time URL and file scanning in a single Chrome extension, offering users a more complete and effective security solution.

Cyber attackers often combine malicious web pages and infected downloads to bypass traditional protection layers. For instance, a phishing site may trick the user into downloading a disguised executable or document, making it essential to have a dual-layered defense system. The lack of such integrated tools inspired the creation of SafeSurf, which offers proactive, real-time alerts using trusted threat intelligence platforms. Additionally, SafeSurf empowers users by allowing them to control blacklist and whitelist settings, adding a customizable touch to their browsing safety.

1.2 Problem Statement

Current security extensions lack the capability to effectively combine both URL scanning and downloaded file scanning in a unified solution. Many security tools today focus either on scanning URLs or on scanning downloaded files, but they typically do not integrate both functions. URL scanning identifies potentially dangerous websites, while file scanning looks for threats in downloaded files, such as viruses or malware. However, no solution currently offers real-time, integrated scanning for both elements together, leaving gaps in overall protection.

This gap in functionality leaves users vulnerable to multi-vector threats, increasing the risk of cyber-attacks, data breaches, and system compromises. Malicious websites and downloaded files often work together to launch attacks. For example, a harmful website might lead to the download of a malicious file, which could infect the system. Tools that only scan URLs or only scan files miss the full picture. Users could be exposed to threats like drive-by

downloads or exploit kits, where both website and file components play a role in the attack. This lack of integrated protection increases the risk of attacks going undetected. **The need for an all-in-one security extension that can scan both URLs and downloaded files simultaneously is critical to enhancing user protection.** An integrated solution that scans both URLs and files in real-time would offer more comprehensive security. Such a tool could block access to dangerous sites and scan any downloaded files for threats.

1.3 Project Objectives

The main goal of the SafeSurf Chrome extension is to provide a secure, real-time protection mechanism for users while browsing the web. This is achieved by integrating URL and file download scanning into a single, lightweight, and efficient browser extension. The following are the key objectives of this project:

1. **Real-Time Threat Detection:** To detect and alert users about potentially harmful URLs and downloadable files in real time using the VirusTotal API.
2. **Dual-Function Security:** To offer an integrated approach to web security by combining both URL and file scanning, which is often missing in traditional browser security tools.
3. **Lightweight Extension:** To ensure that the extension remains lightweight and does not slow down the browsing experience, making it suitable for daily use.
4. **User-Friendly Interface:** To develop a simple and intuitive interface where users can manage threat reports, whitelist trusted sites, and get real-time alerts.
5. **Customizable Security Settings:** To allow users to configure whitelists and blacklists based on their preferences, adding flexibility to the extension.
6. **Leverage Threat Intelligence:** To make use of reputable threat intelligence platforms like VirusTotal and Google Safe Browsing to enhance detection accuracy and stay updated with the latest threats.
7. **Encourage Proactive Web Safety:** To educate and empower users by proactively notifying them of risky interactions online rather than reacting after a threat has been executed.

1.4. Project Report Organization

This project report is structured into the following chapters to present a comprehensive view of the SafeSurf Chrome Extension:

- **Chapter 1: Introduction** – Provides background information, motivation behind the project, the problem statement, objectives, and the overall structure of the report.
- **Chapter 2: Literature Survey** – Discusses existing tools and technologies related to web security and highlights their limitations which SafeSurf aims to overcome.
- **Chapter 3: Software & Hardware Specifications** – Lists the software and hardware requirements for implementing the SafeSurf extension. Also includes detailed project objectives and the motivation behind the solution.
- **Chapter 4: Proposed System Design** – Describes the system's overall architecture, the methods used, and includes diagrams such as use case, activity, sequence diagrams, and the technology stack involved.
- **Chapter 5: Implementation & Testing** – Details the actual implementation of the extension, screenshots of the user interface, and testing methods and results.
- **Chapter 6: Conclusion & Future Scope** – Summarizes the outcomes of the project and proposes possible enhancements for future versions of SafeSurf.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing Work

Cyber threats are evolving, with malicious websites often leading to harmful file downloads. However, most existing browser security extensions focus on either **URL scanning** or **file scanning**, leaving users vulnerable to multi-vector attacks.

Some well-known extensions include:

- **PIXM** – Uses AI to detect phishing attacks in real-time but lacks file scanning.
- **Phish Detector** – Identifies phishing websites but does not analyze downloaded files.
- **AI Phishing Assistant** – Machine learning-based phishing detection but no file scanning.
- **OPSWAT File Security** – Scans downloaded files for malware but does not monitor URLs.
- **NoPhish Phishing detection:** Detect phishing websites in real-time using machine learning techniques but does not analyze downloaded files.

Need for SAFESURF

To address this gap, **SAFESURF** integrates **both URL and file scanning** in a single browser extension. It uses **VirusTotal API** to protect users in real-time. The extension provides **instant alerts, customizable whitelisting, and seamless protection** against phishing, malware, and other cyber threats, ensuring a safer browsing experience.

2.2 Limitations of Existing Work

Despite the availability of these security tools, existing browser extensions have several limitations that leave users vulnerable to modern cyber threats. The primary drawback is that most security extensions focus on **either URL scanning or file scanning**, failing to provide a **unified, real-time protection mechanism**. Malicious websites often lead to harmful file downloads, but without an integrated security approach, users remain exposed to **multi-stage cyberattacks** such as drive-by downloads and phishing-based malware infections.

Moreover, many of these tools rely heavily on **static blacklists**, which can quickly become outdated as cybercriminals continuously generate new phishing domains and malware variants. This limits their effectiveness against **zero-day threats** and newly emerging attack techniques. Additionally, some security extensions require **manual scanning** instead of offering automated, real-time protection, making them less efficient for proactive threat detection.

Another significant limitation is the **lack of user customization options**. Many existing solutions do not allow users to **whitelist trusted websites or adjust scanning settings**, which can lead to unnecessary warnings or false positives. Furthermore, some tools may introduce **performance overhead**, slowing down browsing speed due to inefficient scanning mechanisms.

Given these challenges, there is a need for a **comprehensive security extension** that integrates **real-time URL and file scanning**, leveraging threat intelligence APIs and AI-driven threat detection to offer **seamless, automated protection** against phishing, malware, and other web-based attacks.

Table: 2.2 Differences

Extension Name	URL Scanning	File Scanning	Real-time Protection	Open Source	Limitation
PIXM	✅	❌	✅	❌	Detects phishing in real-time but doesn't scan downloaded files.
PhishDetector	✅	❌	❌	❌	Identifies phishing websites, no real-time protection or file analysis.
AI Phishing Assistant	✅	❌	✅	❌	Machine learning-based detection but no file scanning.
OPSWAT File Security	❌	✅	✅	❌	Scans only files for malware; does not monitor or analyze URLs.
NoPhish	✅	❌	✅	❌	Detects phishing URLs with ML but no scanning of downloaded content.

The above table 2.2 explains the differences between the current existing tools.

CHAPTER 3
SOFTWARE & HARDWARE SPECIFICATIONS

3.1 Software Requirements

This section includes the software tools, technologies, and APIs used in the development and deployment of the SafeSurf Chrome Extension:

- **Operating System:** Windows 10 or later / Chrome OS / Linux
- **Web Browser:** Google Chrome (latest version)
- **Languages Used:** HTML, CSS, JavaScript
- **APIs:**
    - VirusTotal API (for threat detection)
- **Code Editor:** Visual Studio Code
- **Version Control:** GitHub (for collaborative development and version management)
- **API function calls:** Javascript API validation and function calls
- **Chrome Developer Tools:** For debugging and extension testing

These software tools are necessary to build a functional, responsive, and secure extension for web threat detection.

3.2 Hardware Requirements

To ensure the smooth operation of the SafeSurf extension during development and user testing, the following minimum and recommended hardware specifications were considered:

- **Processor:**
    - Minimum: Intel Core i3
    - Recommended: Intel Core i5 or higher
- **RAM:**
    - Minimum: 4GB
    - Recommended: 8GB or higher
- **Storage:**
    - Minimum: 250MB of free space (for project files and supporting tools)
- **Internet Connectivity:** Required for Realtime scanning.

CHAPTER 4
PROPOSED SYSTEM DESIGN

4.1 Proposed Methods

SafeSurf is a lightweight and efficient Chrome extension that provides dual-layer protection by scanning both URLs and downloaded files in real-time. It utilizes trusted threat intelligence sources such as VirusTotal and Google Safe Browsing to protect users during web browsing.

Key Methodologies:

1. Real-Time Monitoring

- Continuously tracks visited URLs and downloaded files.
- Uses Chrome's webRequest and downloads APIs to detect activity.

2. Threat Detection Using APIs

- **URL Scan:** Captures and sends URLs to VirusTotal. Alerts the user instantly if flagged.
- **File Scan:** Computes SHA256 hash of downloaded files and checks against VirusTotal.

3. User Alerts & Notifications

- Displays alert popups if any threat is detected, preventing interaction.

4. Whitelist/Blacklist Support

- Users can define trusted and blocked URLs to refine protection.

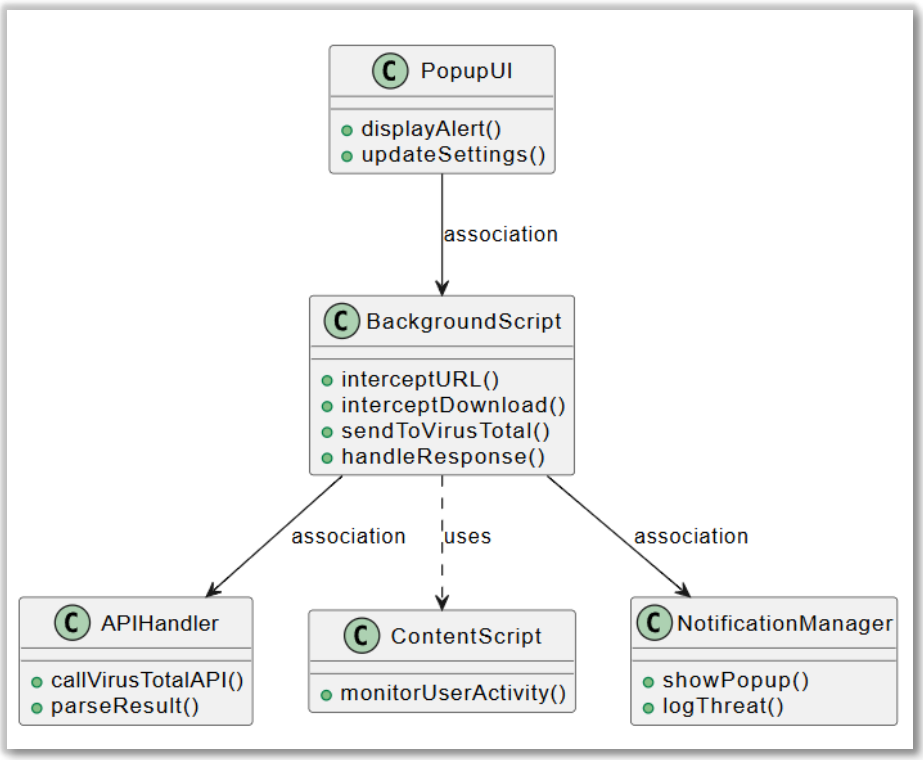
5. Lightweight and Responsive

- Asynchronous scanning to ensure smooth browsing.
- Efficient resource usage to avoid lag.

6. Component-Based Architecture

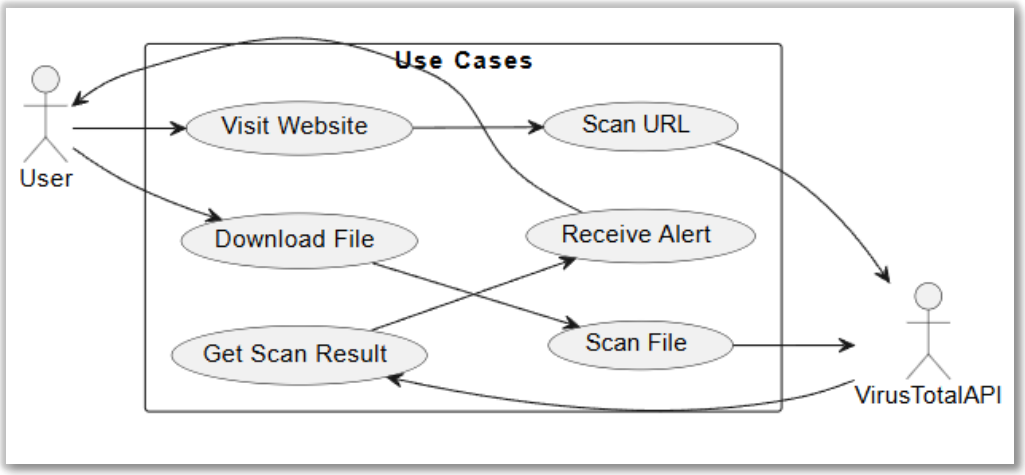
- **Content Scripts** – Capture browsing data.
- **Background Scripts** – Handle threat checks and logic.
- **Popup UI** – User interactions and settings.

4.2 Class Diagram



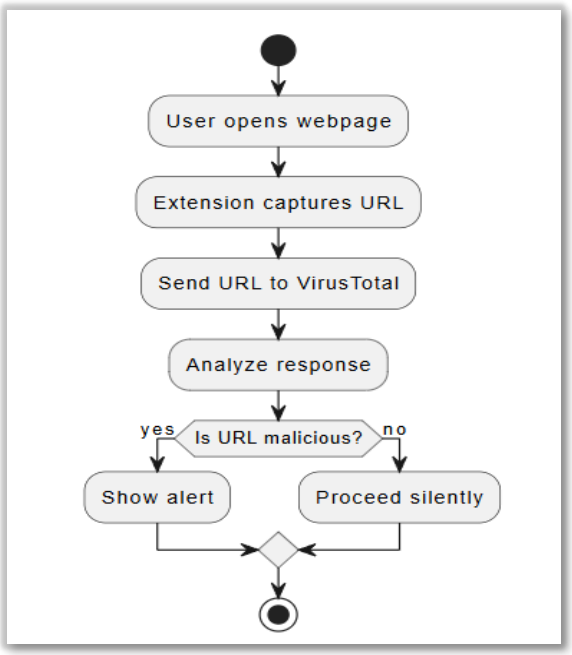
The 4.2 class diagram explains about Chrome extension where Background Script coordinates URL/download interception and threat detection. It interacts with UI, API, content, and notification components.

4.3 Use case Diagram



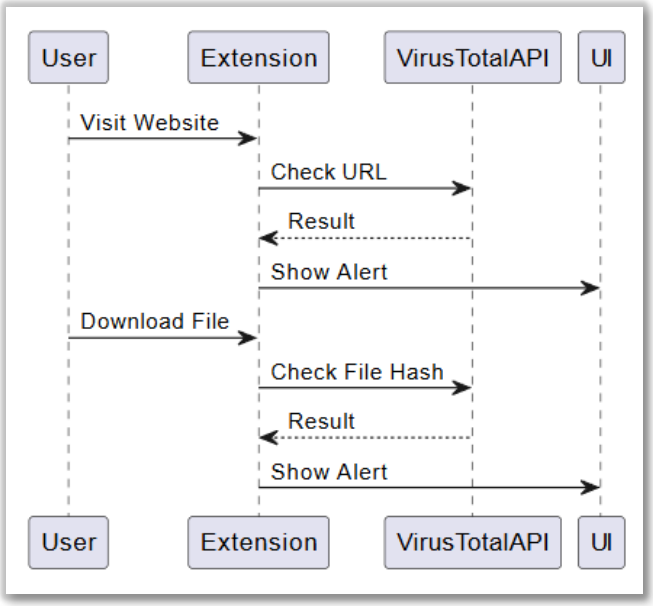
The 4.3 Use case Diagram explains how a user interacts with a Chrome extension to scan URLs and files. It also illustrates communication with the VirusTotal API to detect threats and return scan results.

4.4 Activity Diagram



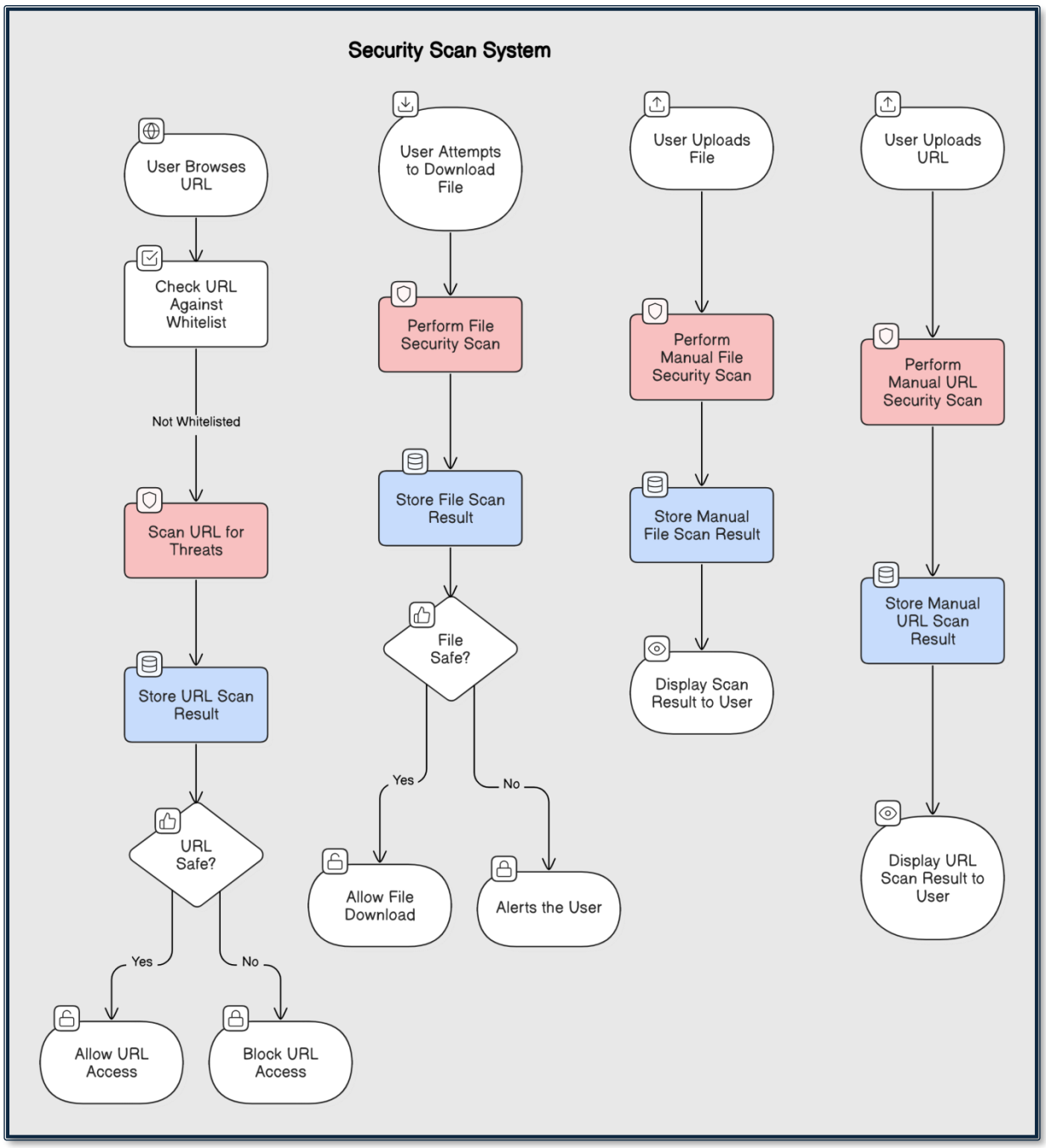
The 4.4 Activity Diagram explains how the extension captures a webpage URL, sends it to VirusTotal, analyzes the response, and either alerts the user or proceeds silently based on threat detection.

4.5 Sequence Diagram



This 4.5 Sequence diagram illustrates the extension's workflow. When a user visits a website or downloads a file, the extension checks the URL or file hash against the VirusTotal API and displays an alert in the UI based on the results.

4.6 System Architecture



The 4.6 architecture diagram details a security scan system for URLs and files. When a user browses a URL, it's checked against a whitelist; if not whitelisted, it's scanned for threats. Similarly, when a user attempts to download or manually uploads a file, or manually submits a URL, a security scan is performed. The system then stores the scan results and either allows or blocks access/download, or alerts the user based on the safety assessment.

4.7 Technology Description

SafeSurf is developed using modern web technologies and Chrome extension APIs to provide real-time threat detection while maintaining performance and usability.

Chrome Extension APIs

- **WebRequest API**: Intercepts and monitors outgoing web requests to capture URL traffic.
- **downloads API**: Detects and handles file download events.
- **runtime & storage APIs**: Manages communication between scripts and stores local preferences like whitelist/blacklist.

Languages & Tools

- **JavaScript**: Core logic for API interaction and asynchronous scanning.
- **HTML/CSS**: For designing the extension's popup interface and user alerts.
- **Manifest V3**: The structure file defining permissions and background script configuration.

External Security Services

- **VirusTotal API**: Used to analyze both URLs and file hashes (SHA256) using a large database of known threats.

Modular Architecture

- **Content Script**: Detects user interaction with pages.
- **Background Script**: Manages the scanning logic and API integration.
- **Popup UI**: Interfaces with the user to show details.

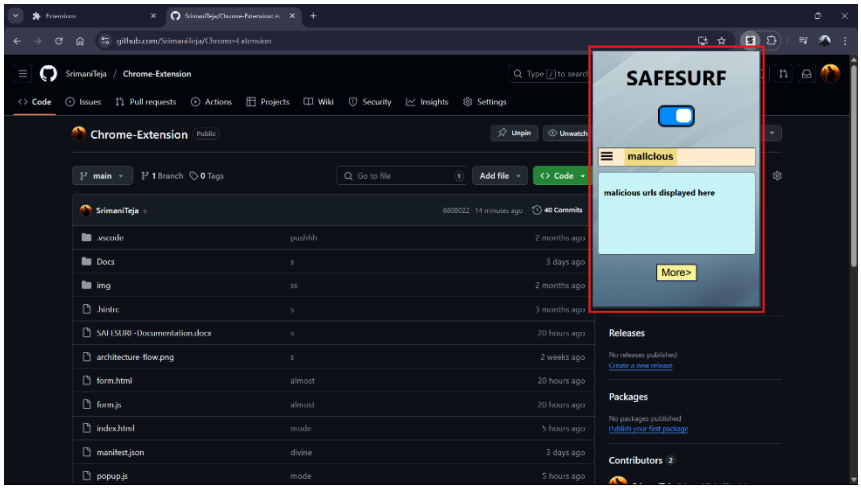
Benefits of This Stack

- Lightweight and fast with minimal impact on browser performance.
- Real-time detection powered by globally updated threat databases.
- User-friendly and customizable through built-in storage.
- Easily extendable to support other browsers and additional features.

CHAPTER 5
IMPLEMENTATION & TESTING

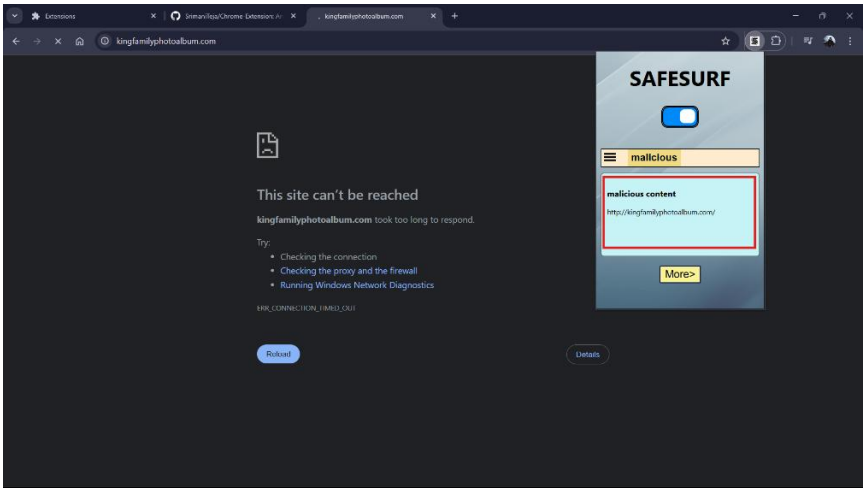
5.1 Frontend Screenshots

Fig: a) SafeSurf Window



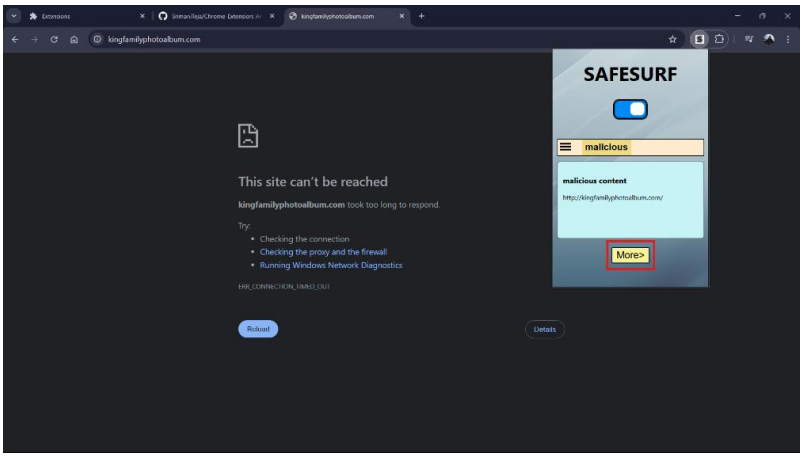
The Fig: a display our SafeSurf extension window.

Fig: b) URL Scan



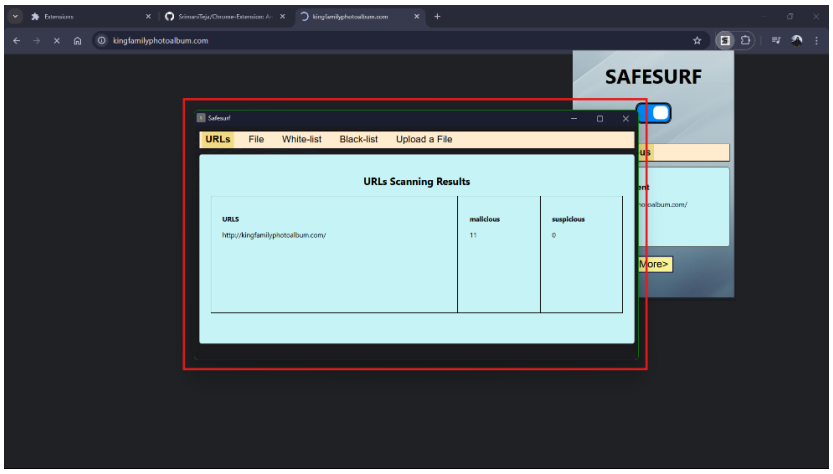
The Fig: b shows the blocked URLs that are been browsed by the user and it is displayed right in the extension's body.

Fig: c) More Button



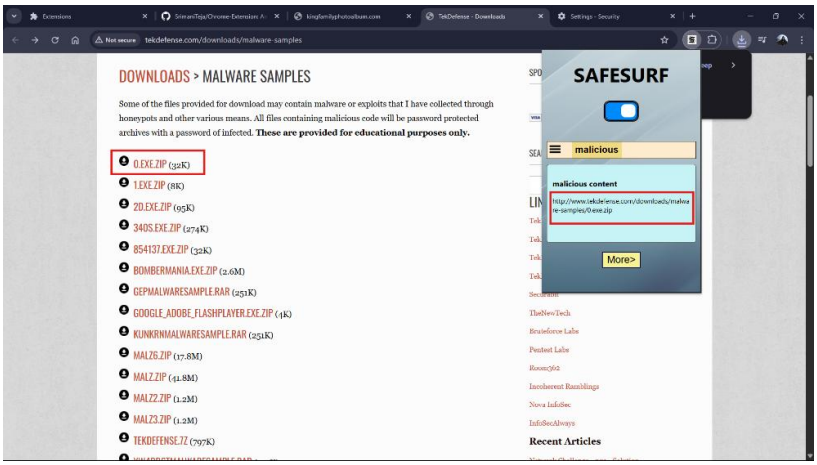
The Fig: c shows about the option "More" that we have provided in the extension

Fig: d) More Features



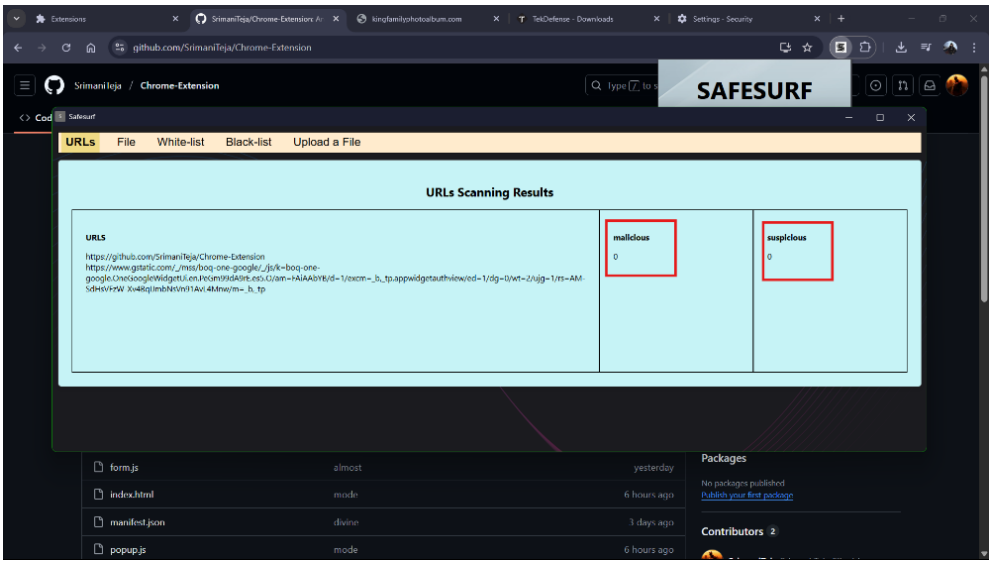
The Fig: d displays the different options that are available when the more button is clicked like URL history, File History, Whitelist, Blacklist and File Upload

Fig: e) File Scan



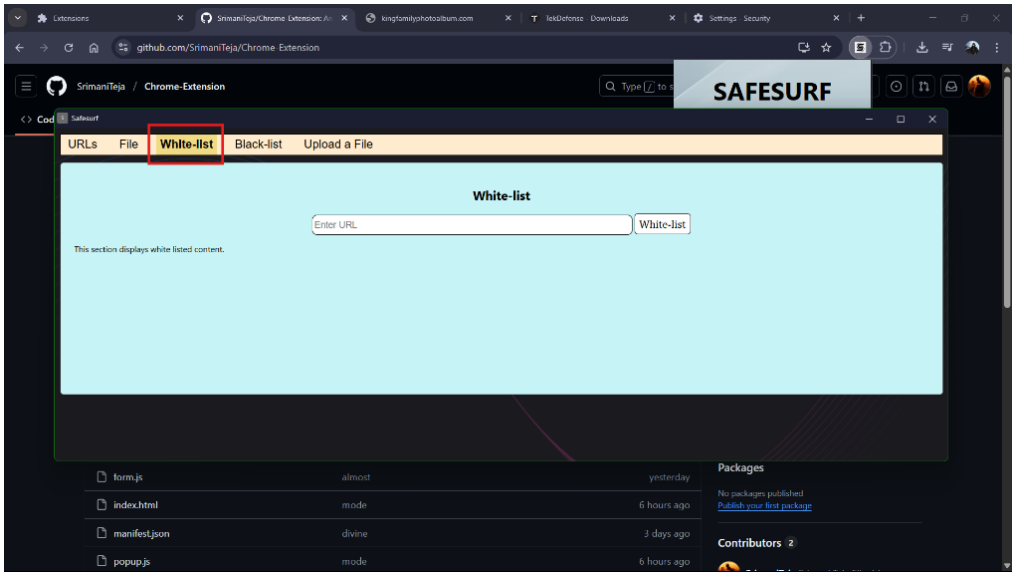
The Fig: e shows that whenever the user tries to download a malicious file the extension blocks it and displays malicious file URL in the extension's body.

Fig: f) URL Results



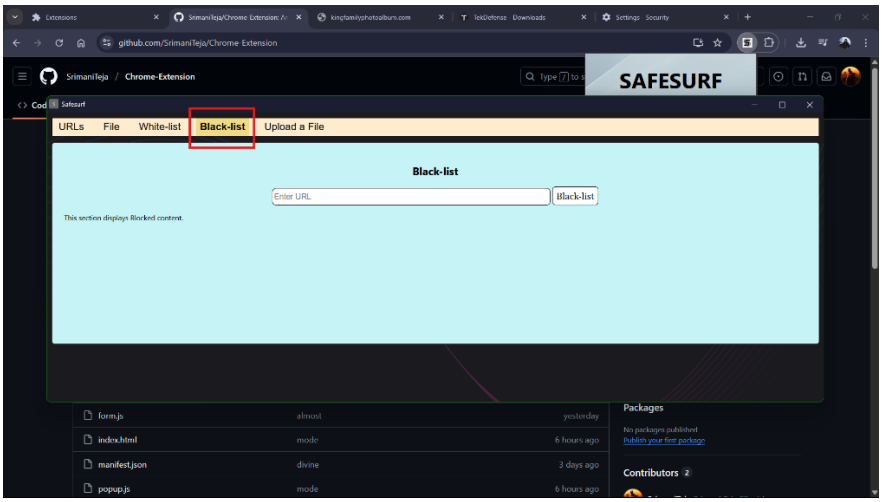
The Fig: f shows the results of the URL's scan history with there malicious and suspicious scores according to the VirusTotal database.

Fig: g) Whitelisting option



The Fig: g shows the feature of manually whitelisting a site by the user.

Fig: h) Blocklisting option



The Fig: h displays the Blocklisting feature that is present in the more button and the user can manually block URLs here.

5.2 Backend Code Snippets

Fig: i) Manifest

```
{
  "name": "Safe-Surf",
  "description": "Base Level Extension",
  "version": "1.0",
  "manifest_version": 3,
  "permissions": [
      "activeTab",
      "tabs",
      "webRequest",
      "webNavigation",
      "management",
      "scripting",
      "downloads",
      "storage"
  ],"host_permissions": [
      "<all_urls>"
  ],
  "action": {
    "default_popup": "index.html"
    , "default_icon": "./img/logo2.png"
  },
  "background": {
    "service_worker": "service.js",
    "type": "module"
  }
}
```

The Fig: i shows the manifest.json file that has all the permissions needed for URLs, active tabs, web request, web navigation and more.

Fig: j) Gather White and Black

```
13    chrome.storage.local.set({ data: safetydata }, () => {
14      console.log("data:",safetydata);
15    });
16    chrome.storage.local.set({ blocked: blockedurls}, () => {
17      console.log("blocked",blockedurls);
18    });
```

The Fig: j shows the code that collects the whitelist and blacklist data from the local storage which is given by the user.

Fig: k) ID Generation

```
148  function truesafe(sendurl){
149
150    allurls.push(sendurl)
151
152    const encodedParams = new URLSearchParams();
153    encodedParams.set('url', sendurl);
154
155    const url = 'https://www.virustotal.com/api/v3/urls';
156    const options = {
157      method: 'POST',
158      headers: {
159        accept: 'application/json',
160        'x-apikey': '38f63e942fdc9f5027407d6b08351e43ac7e11e27f432c1fb76b08ca4977b205',
161        'content-type': 'application/x-www-form-urlencoded'
162      },
163      body: encodedParams
164    };
165
166    fetch(url, options)
167      .then(res => res.json())
168      .then(json =>{
169        sendid=json.data.id
170        console.log(sendid)
171        const url1 = 'https://www.virustotal.com/api/v3/analyses/'+sendid;
172        const options1 = {
173          method: 'GET',
174          headers: {
175            accept: 'application/json',
176            'x-apikey': '38f63e942fdc9f5027407d6b08351e43ac7e11e27f432c1fb76b08ca4977b205'
177          }
178        };
```

The Fig: k's code explains about the generation of unique ID for each URL which is browsed by the user.

Fig: l) Fetching Results

```
        fetch(url1, options1)
          .then(res => res.json())
          .then(json => {
            console.log(json.data.attributes.stats)
            safetydata.push(json.data.attributes.stats.malicious)
            safetydata.push(json.data.attributes.stats.suspicious)
            if(json.data.attributes.stats.malicious>0 || json.data.attributes.stats.suspicious>0){
              chrome.action.openPopup();
              blockedurls.push(sendurl)
              chrome.storage.local.set({ blocked: blockedurls}, () => {
                console.log("blocked",blockedurls);
              });

            }
            chrome.storage.local.set({ data: safetydata }, () => {
              console.log("data:",safetydata);
            });
          })
          .catch(err => console.error(err));
      })
      .catch(err => console.log(err));
  }
```

The Fig: l's code is about fetching the URL score(result) by sending the unique ID that is been generated to VirusTotal.

Fig: m) Syncing the variables

```
// Function to update data in chrome.storage.local
function updateStoredData() {
  chrome.storage.local.set({ message: allurls }, () => {
  });
}

// Update data whenever the popup is opened
chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
  if (message.action === "updateData") {
      updateStoredData();
      sendResponse({ status: "Data updated!" });
  }
});



chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
    if (message.action === "whitelist") {
      let updatedValue = message.newValue;
      whitelist.push(updatedValue)
    }
    else if(message.action === "blacklist") {
      let updatedValue = message.newValue;
      blacklist.push(updatedValue)
    }

    console.log("whitelist",whitelist,"blacklist",blacklist)
});
```

The Fig: m's is about syncing the variables by updating the data.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

SafeSurf offers a modern and efficient solution for secure web browsing by combining real-time URL and file scanning into a single Chrome extension. By leveraging crowd-sourced threat intelligence APIs like VirusTotal and Google Safe Browsing, the extension can detect threats before they reach the user. With its asynchronous design, SafeSurf operates smoothly in the background without compromising user experience. Its modular architecture and lightweight design make it both scalable and highly maintainable.

Future Scope:

Cross-Browser Compatibility: Extend support to Firefox, Microsoft Edge, and other Chromium-based browsers.

User Reporting System: Allow users to manually report suspicious URLs or files to improve community defense.

Scheduled Scans: Add a feature to periodically scan saved URLs or downloaded files in background.

Enhanced Whitelist/Blacklist Controls: Introduce category-based filtering and auto-suggestion for safe domains.

REFERENCES

1. Press , A .: US official : Hackers targeted election systems of 20 states . [Accessed on 07.09.2022] (Retrieved from https://apnews.com/article/c6f67fb36d844f28bd18a522811bdd18)
2. IBM : X - Force Threat Intelligence Index 2022. [Accessed on 09.09.2022] (Retrieved from https://www.ibm.com/downloads/cas/ADLMYLAZ)
3. IBM : X - Force Threat Intelligence Index 2021. [Accessed on 07.09.2022] (Retrieved from https://www.ibm.com/security/data-breach/threat-intelligence)
4. APWG : Phishing Activity Trends Report . [Accessed on 07.09.2022] (Retrieved from https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf)
5. Secure , V .: Phishers ' Favorites 2021 Year - in - Review . [Accessed on 13.09.2022] (Retrieved from https://shorturl.at/lvzG.J)
6. PhishTank : Statistics about phishing . [Accessed on 26.08.2022] (Retrieved from https://www.phishtank.com/stats.php)
7. Kirchner , A. , Signorino , C.S .: Using support vector machines for sur- vey research . Survey Practice 11 (1) (2018) . https://doi.org/10.29115/SP-2018-0001
8. Hastie , T. , Tibshirani , R. , Friedman , J.H. , Friedman , J.H .: The Elements of Statistical Learning : Data Mining , Inference , and Prediction vol . 2 , pp . 282-283 . Springer , New York , NY (2017)
9. Gao , G. , Wang , M. , Huang , H. , Tang , W .: Agricultural irrigation area prediction based on improved random forest model (2021) . https://doi.org/10.21203/rs.3.rs-156767/v1
10. Alkhalil , Z. , Hewage , C. , Nawaf , L. , Khan , I .: Phishing attacks : A recent comprehensive study and a new anatomy . Frontiers in Computer Science 3 , 563060 (2021)
11. Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., Wang, Y.: An effective detection approach for phishing websites using url and html features. Scientific Reports 12(1), 1-19 (2022)
12. Rana, S., Aksoy, A.: Automated fast-flux detection using machine learning and genetic algorithms. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1-6 (2021). https://doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484614

13. Prettejohn, M.: Netcraft Extension (Anti-phishing toolbar). [Accessed on 07.09.2022] (Available at: https://www.netcraft.com/apps/browser/). https://www.netcraft.com/apps/browser/
14. Lin, Y., Liu, R., Divakaran, D.M., Ng, J.Y., Chan, Q.Z., Lu, Y., Si, Y., Zhang, F., Dong, J.S.: Phishpedia: a hybrid deep learning based approach to visually identify phishing webpages. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 3793-3810 (2021)
15. M., M.: PhishDetector - True Phishing Detection. [Accessed on 07.09.2022] (Retrieved from https://chrome.google.com/webstore/detail/phishdetector-true-phishi/kgecldbalfgmgelepbblodfoogmjdgmj)
16. Digitaldream: Cascaded Phish Detector. [Accessed on 07.09.2022] (Retrieved from https://chrome.google.com/webstore/detail/cascaded-phish detector/pfngencjknacaakdekdhjfgoalpjnini)
17. Alam, M.N., Sarma, D., Lima, F.F., Saha, I., Ulfath, R.-E., Hossain, S.: Phishing attacks detection using machine learning approach. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1173-1179 (2020). https://doi.org/10.1109/ICSSIT48917.2020.9214225
18. Gowtham, R., Krishnamurthi, I.: A comprehensive and efficacious architecture for detecting phishing webpages. Comput. Secur. 40, 23-37 (2014). https://doi.org/10.1016/j.cose.2013.10.004
19. Varshney, G., Misra, M., Atrey, P.K.: A phish detector using lightweight search features. Computers & Security 62, 213-228 (2016). https://doi.org/10.1016/j.cose.2016.08.003
20. Mohammad, R.M.A., McCluskey, L., Thabtah, F.: Phishing Websites Data Set. Data retrieved from UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/phishing+websites (2015-2020)

PLAGIARISM REPORT

Chapter No.	Chapter Name	Number of words	Plagiarism(%)
1	Abstract	198	0%
2	Introduction	802	0%
3	Literature Survey	492	7%
4	Software & Hardware Specs	179	0%
5	Proposed System Design	174	0%
6	Implementation & Testing	1870	18%
7	Conclusion & Future Scope	132	0%
Total		**3847**	**8%**

Note: "DupliChecker" plagiarism tool is used for checking chapter-wise plagiarism.

DupliChecker-"https://www.duplichecker.com/

APPENDIX

The entire source code of this project is available on GitHub - **https://github.com/SrimaniTeja/Chrome-Extension**